Escuela
Politécnica
Superior

Estudio de estructuras con retardo de grupo negativo en tecnología microstrip

Máster Universitario en Ingeniería de Telecomunicación

Trabajo Fin de Máster

Autora:
Margarita Martínez Coves
Tutores:
Miguel Ángel Sánchez Soriano
Stephan Marini

Universitat d'Alacant
Universidad de Alicante

Septiembre 2020

Estudio de estructuras con retardo de grupo negativo en tecnología microstrip

Autora
Margarita Martínez Coves

Tutores
Miguel Ángel Sánchez Soriano
Stephan Marini
Departamento de Física, Ingeniería de Sistemas y Teoría de la Señal

Máster Universitario en Ingeniería de Telecomunicación

Escuela
Politécnica
Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Septiembre 2020

Resumen

Los sistemas de telecomunicación que operan en frecuencias de microondas han proliferado mucho en los últimos años. Desde los sistemas de navegación por satélite, pasando por aquellas comunicaciones inalámbricas como WLAN o WMAX o incluso algunas de las bandas reservadas para la telefonía móvil trabajan a dichas frecuencias.

Por todo ello es de gran interés para el avance de las telecomunicaciones disponer de métodos y dispositivos que permitan una comunicación eficiente a frecuencias de microondas. Una de las propuestas que actualmente se baraja para aumentar la eficiencia de los amplificadores, minimizar el estrabismo del haz en sistemas de arrays de antenas, reducir el retardo de las líneas o mejorar el ancho de banda son los dispositivos con retardo de grupo negativo.

En concreto, en este trabajo se estudia una estructura concreta capaz de producir una respuesta con retardo de grupo negativo. Sin embargo, el análisis paramétrico de ciertas estructuras no siempre es trivial, por lo que para poder llevar a cabo el estudio se desarrolla e implementa un sistema de cómputo para obtener un análisis paramétrico por fuerza bruta.

Mediante dicho sistema se analiza la topología propuesta y se caracteriza para obtener respuestas con dos y tres bandas con retardo de grupo negativo, permitiendo así también diseñar dispositivos muy interesantes de cara a sistemas de comunicación que utilicen dos bandas de frecuencia distinta, pues aunaríamos en un solo dispositivo el filtro de ambas.

En conclusión, el presente Trabajo Final de Máster (TFM) expone los principios de la ingeniería de microondas y los aplica al desarrollo de un sistema de procesado y análisis de topologías o estructuras complejas.

Agradecimientos

Este trabajo no habría sido posible sin la inestimable ayuda de mis tutores: Miguel Ángel y Stephan. En concreto, me gustaría agradecer personalmente a Miguel Ángel, que no sólo ha sido un apoyo académico para la realización de este trabajo, sino también una gran inspiración como profesional. Gracias por todo el tiempo invertido durante la elaboración de este TFM, las ideas, las correcciones y los mensajes de ánimo. Gracias a los dos por estos últimos días, pero sobre todo muchísimas gracias por darme la oportunidad de aprender con vosotros de este ámbito de la ingeniería.

Anyone who stops learning is old, whether at twenty or eighty.
Anyone who keeps learning stays young.

Henry Ford (1843-1947)

Índice general

Índice de figuras

Índice de tablas

1. Introducción y motivación

Los sistemas de telecomunicación han proliferado mucho en los últimos siglos. Desde los primeros sistemas telegráficos en 1794 hasta las modernas comunicaciones ópticas a partir de 1973, pasando por todas aquellas comunicaciones *wireless*, sin un medio guiado. La carrera por aumentar la velocidad, fiabilidad y alcance de nuestras comunicaciones nos ha llevado a desarrollar técnicas y tecnologías muy variadas. Una posible clasificación de los sistemas de telecomunicación, se fundamenta en su frecuencia de funcionamiento. Concretamente, este TFM se centra en aquellos que trabajan en frecuencias de microondas, entendiendo como tales aquellas ondas electromagnéticas cuya longitud de onda se ubica entre 1 m y 1 mm (300 y 300000 MHz) [1].

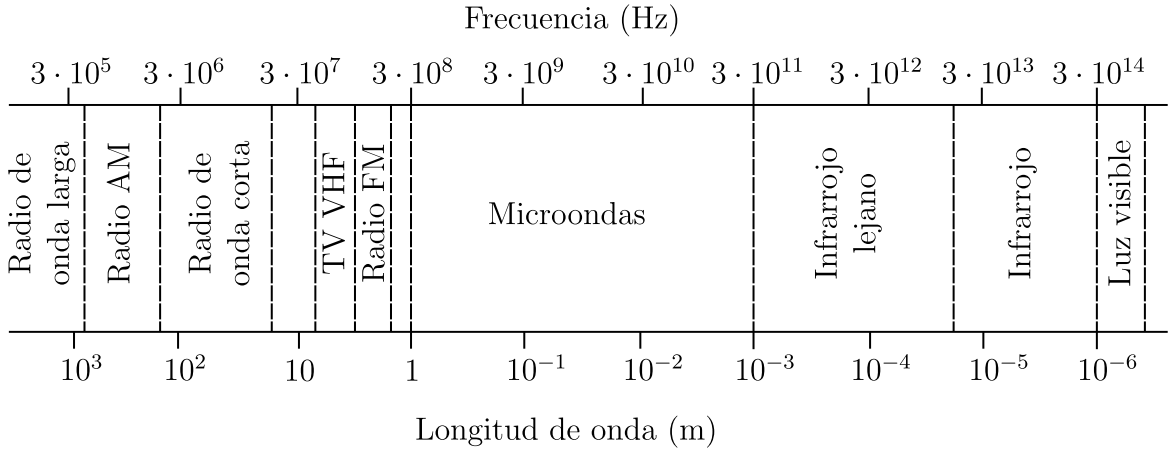


Figura 1.1: El espectro electromagnético (Traducción de la Figura 1.1 de [1])

Dentro de este rango de frecuencias, y concretamente en las bandas situadas entre 1 y 5 GHz, encontramos diversas aplicaciones como sistemas radar para servicios de radiolocalización o localización y navegación por satélite GALILEO [2]. A nivel de conexionado de redes, también se destinan diversas bandas alrededor de los 2,4 y 5 GHz para aplicaciones Wireless Local Area Network (WLAN) [3]; y para telefonía móvil también se reservan y subastan múltiples bandas para diferentes generaciones de comunicaciones móviles.

Teniendo presente estos datos, es de gran interés para la sociedad disponer de métodos y dispositivos que permitan una correcta y eficiente comunicación en frecuencias de microondas. En base a esto, durante los últimos años se han comenzado a desarrollar dispositivos con retardo de grupo negativo (de ahora en adelante NGD). Los objeti-

vos de dichos dispositivos pueden ser: incrementar la eficiencia de los amplificadores de prealimentación lineal, minimizar el estrabismo del haz en sistemas de arrays de antenas, reducir el retardo de las línea o mejorar el ancho de banda [4].

1.1. Estado del Arte

En 1954 Sommerfeld [5] y en 1960 Brillouin [6], publicaron sus trabajos acerca del comportamiento de la luz al propagarse en la región electromagnética de medios dispersivos con índice de refracción n. Teóricamente, en la región donde la línea espectral se define como una línea de absorción, la velocidad de grupo ($v_g(\omega)$ en la Ecuación (1.1)) puede ser superior a la velocidad de la luz en el vacío, c, e incluso negativa [7]. Este fenómeno de velocidad de grupo negativa fue confirmado en 1970 por Garret y McCumber [8], demostrando así la posibilidad de propagar un pulso Gaussiano con un retardo de grupo negativo, donde el retardo de grupo $\tau_g(\omega)$ queda definido mediante la Ecuación (1.2)

$$v_g(\omega) = \frac{c}{\mathrm{Re}[n(\omega)] + \mathrm{Re}[dn(\omega)/d\omega]} \quad \text{(m/s)}, \tag{1.1}$$

$$\tau_g(\omega) = -\frac{\partial\phi(\omega)}{\partial\omega} \quad \text{(s)}. \tag{1.2}$$

Múltiples investigaciones, tanto teóricas como experimentales, han sido llevadas a cabo desde entonces para concretar e ilustrar este fenómeno: basadas en el análisis de medios con una función de transferencia con doble pico de ganancia (*gain-doublet*) [9, 10]; constatando que los pulsos electromagnéticos pueden propagarse de forma superlumínica en aquellos medios que permiten la propagación sin dispersión [11]; o demostrando la posibilidad de incorporar una velocidad de grupo negativa en líneas de transmisión *left-handed* o con índice de refracción negativo [12, 13].

El fenómeno de retardo de grupo negativo puede parecer en un principio estar contradiciendo el principio de causalidad, sin embargo este principio físico es respetado ya que tanto el transitorio inicial como final de la señal impulso están limitados a la velocidad del frente de ondas, la cual nunca supera la velocidad de la luz [14, 15]. De hecho, aquellos dispositivos que presentan NGD pueden ser considerados como un predictor ya que la señal a la salida del circuito está adelantada con respecto a la entrada [16].

El diseño de estos dispositivos con NGD se aborda desde distintas perspectivas, que de forma global podemos clasificar en dos grupos: circuitos activos o pasivos.

1.1.1. Dispositivos activos

Desde años 90s, se comenzaron a implementar topologías con retardo de grupo negativo basadas en filtros amplificadores [17, 15]. Estos circuitos activos con NGD están

basados esencialmente en el concepto de filtros transversales (Figura 1.2), los cuales se han utilizado normalmente en la ecualización y la conformación de pulsos. Para conseguir NGD con estos diseños, es necesario seleccionar una ganancia adecuada para cada amplificador distribuido del circuito [18].

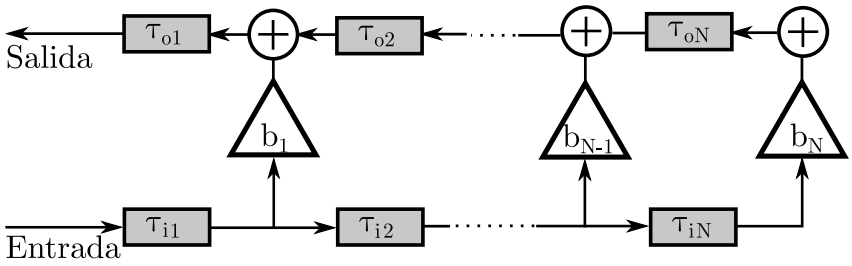


Figura 1.2: Diagrama de un filtro transversal.

En la Tabla 1.1 se exponen los resultados más relevantes hasta la fecha de filtros activos con retardo de grupo negativo.

Ref.	f_0 (GHz)	τ_g (ns)	BW(f_0) (MHz)	S_{21} (dB)
[19]	0,31	−1,52	120	0,29
[20]	1	−10	40	30
[21]	1	−10	40	30
[22]	1	−2,3	150	2,3
[23]	1,03	−2,3	150	1,68
[24]	0,454	−1,52	103	0,069
[18]	1	−1	200	−7,8

Tabla 1.1: Comparativa entre distintas implementaciones de filtros activos con retardo de grupo negativo.

1.1.2. Dispositivos pasivos

Por otro lado, a partir del año 2000 se comienza a estudiar y diseñar dispositivos pasivos capaces de producir NGD. Los circuitos pasivos con retardo de grupo negativo tienen por inconveniente un alto coeficiente de atenuación proporcional al NGD conseguido, es por ello que desde entonces muchos estudios tienen por meta mejorar esta característica.

Una de las propuestas explotadas para conseguir filtros pasivos con retardo de grupo negativo se basa principalmente en técnicas interferenciales. Inicialmente, este diseño

se propuso para obtener filtros banda eliminada (*bandstop*) compactos y con un amplio ancho de banda [25, 26, 27], sin embargo convenientemente ajustados pueden producir retardos de grupo negativos con atenuaciones aceptables. Estos dispositivos interferenciales se basan en la construcción de dos líneas de transmisión paralelas pero desfasadas 180° (Figura 1.3).

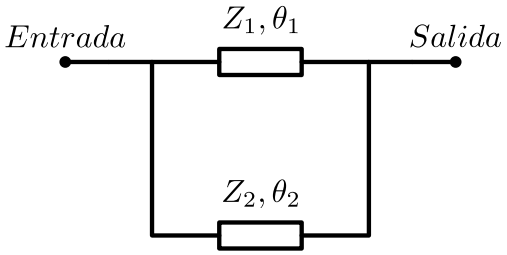


Figura 1.3: Diagrama general de un filtro interferencial.

En la Tabla 1.2 se exponen los resultados más relevantes hasta la fecha de filtros pasivos con retardo de grupo negativo.

Ref.	f_0 (GHz)	τ_g (ns)	BW(f_0) (MHz)	S_{21} (dB)
[28]	2,4	−9	30	−63
[29]	4,5	0,05	3000	−10
[4]	1,962	−7,3	100	−22,65
[4]	1,962	−6,5	200	−35,2
[4]	1,962	−6,5	90	−21,2
[4]	1,962	−6,5	150	−32,94
[30]	2,125	−8,2	40	−37,84
[31]	1,96	−1,1	990	−29,3
[16]	1	−0,8	320	−15

Tabla 1.2: Comparativa entre distintas implementaciones de filtros pasivos con retardo de grupo negativo.

1.1.3. Dispositivos multibanda

Desde 2010 han comenzado a aparecer los primeros estudios sobre dispositivos multibanda, concretamente de dos bandas (*dual-band*) con retardo de grupo negativo. Estos circuitos NGD se basan en los mismos principios que los de una sola banda, sin embargo además de la ya descrita diferencia entre activos y pasivos, podemos diferenciar también aquellos que consiguen dos bandas de funcionamiento debido a una configuración multietapa de aquellos que no. Esta característica otorga mayor independencia entre

las características de una banda y otra, pero también aumenta considerablemente el tamaño del dispositivo final. En la Tabla (1.3) se exponen cronológicamente todos los dispositivos (*dual-band*) hasta la fecha con sus características.

Ref.	f_1 (GHz)	f_2 (GHz)	$\tau_g(f_1)$ (ns)	$\tau_g(f_2)$ (ns)	BW(f_1) (MHz)	BW(f_2) (MHz)	$S_{21}(f_1)$ (dB)	$S_{21}(f_2)$ (dB)
[32][1]	$2,14$	$3,15$	$-7,2$	$-6,5$	$--$	$--$	-30	-30
[33][2]	$1,05$	$2,05$	$-1,16$	$-0,92$	400	605	>0	>0
[34]	$2,14$	$3,50$	-3	$-3,1$	40	80	$-34,2$	$-34,9$
[35][3]	$3,5$	$5,15$	$-4,54$	$-4,20$	120	100	$-47,35$	$-38,8$
[36][3]	$3,38$	$4,7$	$-3,86$	$-3,26$	120	130	$-36,22$	$-34,92$
[37]	$3,50$	$5,20$	-5	-5	200	400	-13	$-19,5$
[38]	$0,667$	$1,377$	$-1,19$	$-1,19$	231	227	$-18,2$	$-18,2$

[1] No se proporcionan datos sobre el ancho de banda, pero se describe como estrecho
[2] Circuito activo
[3] Circuito de dos etapas

Tabla 1.3: Comparativa de los filtros dual-band con NGD hasta la fecha.

El auge de tecnologías *dual-band* como Worldwide Interoperability for Microwave ACcess (WiMAX) o la transmisión WLAN en 2,4 GHz y 5 GHz desde un mismo dispositivo, hacen de los circuitos con retardo de grupo negativo multibanda una línea de investigación más que interesante. Éstos permitirían abaratar costes de producción en dichas tecnologías, mejorar sus prestaciones y facilitar la miniaturización.

2. Marco Teórico

Actualmente las herramientas de computación nos permiten realizar estudios y simulaciones multifísicas exhaustivas. En este sentido, es posible generar modelos que anticipen la respuesta del dispositivo a nivel ideal o bajo condiciones similares a las reales. En este capítulo abordaremos las bases científicas y técnicas de las herramientas y conceptos necesarios para realizar un estudio de estructuras con retardo de grupo negativo en tecnología microstrip.

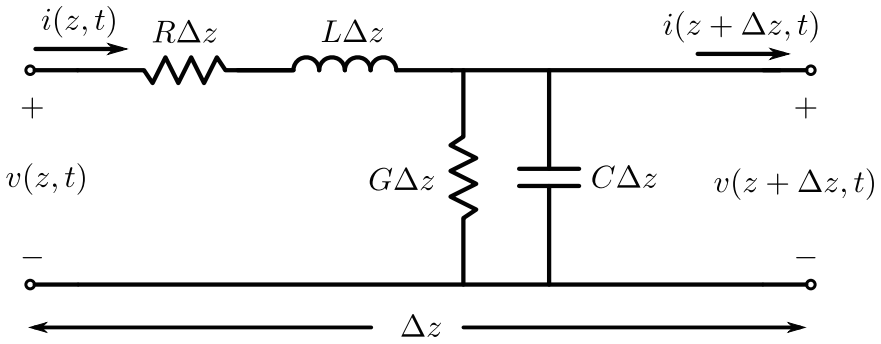
2.1. Ingeniería de Microondas

El diseño y el estudio de dispositivos de Radiofrecuencia (RF) y microondas se asienta sobre las bases de la teoría electromagnética y, por ende, de las ecuaciones de Maxwell. Mientras que para frecuencias bajas es posible modelar el sistema mediante elementos concentrados siguiendo así la teoría de circuitos; para frecuencias altas es necesario optar por un modelo de elementos distribuidos propio de la teoría de líneas de transmisión o, directamente, por la solución teórica del campo electromagnético mediante las ecuaciones de Maxwell [1].

Este requerimiento propio de la alta frecuencia se debe a la relación entre la longitud física (l) y la longitud de onda (λ). Puesto que $\lambda = c/f$, para frecuencias bajas, como los 50 o 60 Hz de la red eléctrica, tenemos longitudes de onda equivalentes al radio de la tierra. Ante tales órdenes de magnitud, podemos asumir que los cambios de la longitud física no afectan en exceso a la fase de la señal a transmitir. Sin embargo, tal y como hemos comentado en el capítulo 1, para las frecuencia de microondas esta longitud de onda va desde 10 cm a 1 mm, por lo que pequeñas variaciones en la longitud física ocasionan cambios significativos en la fase.

2.1.1. Teoría de Líneas de Transmisión

Tal y como se ha comentado en el apartado anterior, la teoría de circuitos no es aplicable para el estudio de de redes de microondas. Una posible solución sería el estudio de los campos eléctrico y magnético mediante la resolución de las ecuaciones de Maxwell, sin embargo es una opción costosa y poco eficiente ya que aporta información, en muchas ocasiones, no necesaria. La teoría de líneas de transmisión tiende un puente entre ambas opciones, pudiendo estudiarse como una extensión de la teoría de circuitos o como una particularización de las ecuaciones de Maxwell [39]. Para definir los conceptos generales necesarios para este TFM partiremos de la teoría de circuitos.



R = Resistencia por unidad de longitud (Ω/m)
L = Inductancia por unidad de longitud (H/m)
G = Conductancia por unidad de longitud (S/m)
C = Capacitancia por unidad de longitud (F/m)

Figura 2.1: Equivalente eléctrico por elementos concentrados de una línea de transmisión

Es posible definir una línea de transmisión mediante el equivalente circuital de la Figura 2.1, que modela una línea de dos conductores, que es el mínimo necesario para que se dé el modo de propagación Transversal Electromagnético (TEM). A partir de los parámetros electromagnéticos (R, L, G y C) podemos determinar la propagación de las ondas en dicha línea. La constante de propagación (γ) queda definida en función de la frecuencia (ω):

$$\gamma = \alpha + j\beta = \sqrt{(R + j\omega L)(G + j\omega C)} \quad (\text{Np/m}), \tag{2.1}$$

donde (α) se corresponde con el coeficiente de atenuación y (β) al coeficiente de fase. Además, la impedancia característica de la línea (Z_0) queda definida por:

$$Z_0 = \sqrt{\frac{R + j\omega L}{G + j\omega C}} \quad (\Omega). \tag{2.2}$$

Por otro lado, puesto que la longitud de onda λ se define como la distancia dos puntos consecutivos que mantengan la misma fase, ésta se puede definir para una línea de transmisión como:

$$\lambda = \frac{2\pi}{\beta} \quad (\text{m}). \tag{2.3}$$

Y, ya que la velocidad de fase o propagación (v_p) es la velocidad a la que un punto de fase constante se propaga, se obtiene la relación:

$$v_p = \frac{\omega}{\beta} = \lambda f \quad (\text{m/s}). \tag{2.4}$$

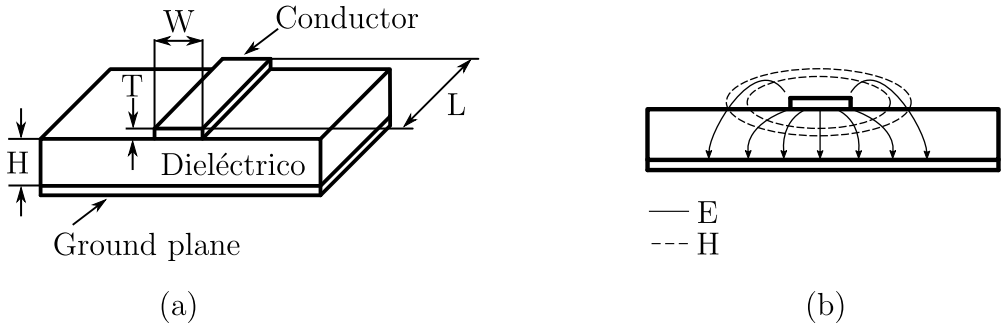
De la Ecuación 2.1 se intuye que en medios sin pérdidas $(\alpha = 0)$ es posible simplificar

las ecuaciones. También, la posibilidad obtener los valores de voltaje y corriente a la entrada y la salida del circuito de la Figura 2.1 aplicando las leyes de Kirchhoff para el análisis de circuitos.

2.1.2. Líneas Microstrip

La tecnología Microstrip nace en 1952 de la mano de Grieg y Engelmann en los labotorios de *ITT Federal Telecommunications* [40]. Esta tecnología nace de la evolución de los sistemas de transmisión coaxial planar y de líneas paralelas durante la Segunda Guerra Mundial [41]. En la actualidad, las líneas Microstrip son una de las tecnologías de líneas de transmisión planar más utilizadas ya que permite optimizar el tamaño del dispositivo y la integración con dispositivos activos y pasivos es sencilla [42].

En la Figura 2.2 se representa esquemáticamente una línea microstrip, indicando en (a) su composición así como las medidas geométricas relevantes y en (b) el comportamiento de las líneas de campo.



W: Ancho del conductor (mm)
L: Longitud del conductor (mm)
T: Espesor del conductor (mm)
H: Espesor del substrato dieléctrico (mm)

Figura 2.2: Representación de una línea Microstrip. (a) Geometría y especificación. (b) Líneas de campo magnético (H) y eléctrico (E)

El análisis de una línea microstrip es algo más complejo que el de la línea de transmisión estudiada en el apartado 2.1.2. Esto se debe a que, a diferencia de otras tecnologías, el medio dieléctrico en el que está sumergido el conductor no es homogéneo: está rodeado por un substrato dieléctrico (de permitividad relativa $\epsilon_r > 1$) y el aire ($\epsilon_r = 1$). Si este diélectrico no estuviese presente, el dispositivo sería una línea de dos conductores compuesta por una tira (*strip*) conductora sobre el plano de tierra (*ground plane*). En ese hipotético caso sería una línea de transmisión TEM simple como la del apartado anterior [42].

Sin embargo, en las líneas microstrip, este doble material dieléctrico provoca que

no todos los campos electromagnéticos estén contenidos en el substrato dieléctrico. Aunque la mayoría de líneas de campo se encuentran en dicha región (Figura 2.2 (b)), algunas de ellas se propagan por el aire, impidiendo así una transmisión TEM pura. La velocidad de fase en el dieléctrico sería $v_p = c/\sqrt{\epsilon_r}$ mientras que en el aire sería igual a la velocidad de la luz c.

El análisis exacto del campo electromagnético en una línea microstrip es complicado. Por suerte, a bajas frecuencias el régimen es quasi-TEM (el comportamiento es prácticamente el mismo que en corriente continua [42]), lo cual nos permite expresar la velocidad de fase y constante de propagación como:

$$v_p = \frac{c}{\sqrt{\epsilon_{r,eff}}} \quad \text{(m/s)}, \tag{2.5a}$$

$$\beta = k_0\sqrt{\epsilon_{r,eff}} \quad \text{(Np/m)}, \tag{2.5b}$$

donde $k_0 = 2\pi f/c$ y $\epsilon_{r,eff}$ es la permitividad efectiva del dieléctrico se puede calcular a partir de la siguiente aproximación:

$$\epsilon_{r,eff} \approx \frac{\epsilon_r + 1}{2} + \frac{\epsilon_r - 1}{2}\sqrt{1 + \frac{12H}{W}}, \tag{2.6}$$

El cálculo de la frecuencia de corte a partir de la cual efectos parásitos aparecen no es objeto de este TFM. A modo de aproximación, se puede estimar que para $H \ll \lambda$ (H inferior al 10% de λ) el régimen de propagación es quasi-TEM y es posible definir la impedancia caracterítica de la línea mediante la siguiente expresión:

$$Z_0 \approx \begin{cases} \dfrac{60}{\sqrt{\epsilon_{r,eff}}} \ln\left(\dfrac{8H}{W} + \dfrac{W}{4H}\right) & \text{si } W/H \leq 1 \\ \dfrac{120\pi}{\sqrt{\epsilon_{r,eff}}} \left[\dfrac{W}{H} + 1,393 + 0,667 \ln\left(\dfrac{W}{H} + 1,444\right)\right]^{-1} & \text{si } W/H \geq 1 \end{cases} \quad (\Omega) \tag{2.7}$$

Donde para obtener el ratio W/H es necesario aplicar las siguientes expresiones:

$$\frac{W}{H} \approx \begin{cases} \dfrac{8e^A}{e^{2A} - 2} & \text{si } W/H \leq 2 \\ \dfrac{2}{\pi}\left[B - 1 - \ln(2B - 1) + \dfrac{\epsilon_r - 1}{2\epsilon_r}\left(\ln(B - 1) + 0,39 - \dfrac{0,61}{\epsilon_r}\right)\right] & \text{si } W/H \geq 2 \end{cases} \tag{2.8}$$

$$A = \frac{Z_0}{60}\sqrt{\frac{\epsilon_r + 1}{2}} + \frac{\epsilon_r - 1}{\epsilon_r + 1}\left(0,23 + \frac{0,11}{\epsilon_r}\right) \tag{2.9a}$$

$$B = \frac{377\pi}{2Z_0\sqrt{\epsilon_r}} \tag{2.9b}$$

Para conocer la longitud de la línea (L) se aplica la siguiente relación:

$$\ell = \frac{\theta}{\beta} \quad \text{(m)} \tag{2.10}$$

donde θ es el desfase en radianes producido por la línea.

Por otro lado, para calcular la atenuación de la línea microstrip es necesario tener en cuenta la atenuación debida al substrato dieléctrico (α_d) y la atenuación debida al conductor (α_c) [43]:

$$\alpha = \alpha_d + \alpha_c \quad \text{(Np/m)}. \tag{2.11}$$

Aunque en muchos casos las pérdidas del dieléctrico son menos relevantes que las debidas al conductor (especialmente si se trabaja a baja frecuencia) ambas pueden ser calculadas mediante las siguientes aproximaciones [44]:

$$\alpha_d = \frac{\pi\epsilon_r}{\lambda\sqrt{\epsilon_{r,eff}}}\frac{(\epsilon_{r,eff} - 1)}{(\epsilon_r - 1)}\tan(\delta) \quad \text{(Np/m)}, \tag{2.12a}$$

$$\alpha_c = \frac{R_s}{Z_0 W} = \frac{\sqrt{\omega\mu_0/2\sigma}}{Z_0 W} \quad \text{(Np/m)}, \tag{2.12b}$$

donde σ es la conductividad eléctrica del conductor y μ_0 la permeabilidad magnética del medio.

2.1.3. Análisis de Redes de Microondas

Tal y como se ha ilustrado a lo largo de este capítulo, el análisis de un dispositivo de microondas puede llegar a complicarse en exceso mediante el estudio del campo electromagnético. Ante esto, existe la posibilidad de analizar estos dispositivos mediante lo que se conoce como Red de Microondas. En la Figura 2.3 se muestra el esquema de una red de microondas de dos puertos, que se corresponde con las analizadas en este TFM, aunque una red puede constar de N-puertos.

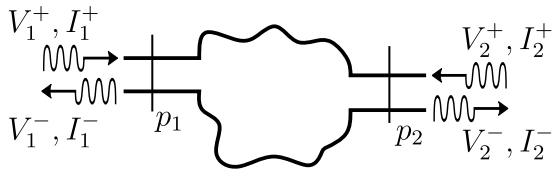


Figura 2.3: Representación general de una red de dos puertos

La ventaja de éste análisis reside en la visión de todo el dispositivo como una matriz que describe su comportamiento:

$$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} \tag{2.13}$$

Donde $[Z]$ sería la matriz de impedancia y es igual a la inversa de la admitancia $[Y]$, mediante la cual también podemos describir la red:

$$\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \tag{2.14}$$

donde cabe destacar que los valores de voltaje y corriente implican a los valores incidentes y reflejados:

$$V_n = V_n^+ + V_n^- \quad \text{(V)} \tag{2.15a}$$
$$I_n = I_n^+ - I_n^- \quad \text{(A)} \tag{2.15b}$$

En este sentido, es complicado medir voltajes y corrientes directamente a frecuencia de microondas debido a la influencia de la fase, tal y como hemos comentado al inicio de este capítulo. Por ello existen también otras matrices que describen el sistema, como la matriz de dispersión o *Scattering* o la matriz de transmisión ABCD. Ya que en el presente TFM se hace uso de éstas dos últimas para obtener la respuesta del sistema, se hará hincapié en ellas.

En cualquiera de los casos, para analizar una red de microondas es imprescindible conocer la impedancia característica (Z_0), constante de propagación (β) y/o longitud eléctrica (ℓ) de los elementos que la conforman. Por suerte, en este TFM se analizan redes con elementos conocidos y esto facilita el proceso.

2.1.3.1. Matriz de Scattering

Para una red de dos puertos donde V_n^+ es la amplitud de la onda incidente y V_n^- de la reflejada, la matriz de Scattering se define como:

$$\begin{bmatrix} V_1^- \\ V_2^- \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} V_1^+ \\ V_2^+ \end{bmatrix} \tag{2.16}$$

De esta forma, si analizamos cada elemento de la matriz $[S]$, se definen como:

$$S_{ij} = \frac{V_i^-}{V_j^+}\Big|_{V_k^+=0 \text{ para } k\neq j} \tag{2.17}$$

por lo tanto: S_{11} : Coeficiente de reflexión del puerto de entrada.
S_{21} : Coeficiente de transmisión.
S_{12} : Coeficiente de transmisión inversa.
S_{22} : Coeficiente de reflexión del puerto de salida.

Cabe destacar la condición expresada en la Ecuación 2.17: todo aquel puerto que no sea sujeto del cálculo del parámetro en cuestión debe estar adaptado en impedancia para evitar reflexiones.

Tal y como se detallará en el Capítulo 3, el dispositivo bajo estudio en este TFM es recíproco. Una red de microondas recíproca no la compone ningún elemento activo o no recíproco como ferrita o plasma [45]. Bajo estas condiciones es posible demostrar que las matrices de impedancia $[Z]$, admitancia $[Y]$ y Scattering $[S]$ son simétricas, es decir: $Z_{ij} = Z_{ji}$, $Y_{ij} = Y_{ji}$ y $S_{ij} = S_{ji}$. Por lo que en una red de dos puertos $S_{21} = S_{12}$.

Por otro lado, si la red bajo análisis no tiene pérdidas es posible demostrar que la matriz de Scattering es unitaria y se cumpliría, para una red de dos puertos, que:

$$|S_{11}|^2 + |S_{21}|^2 = 1 \tag{2.18a}$$
$$|S_{22}|^2 + |S_{12}|^2 = 1 \tag{2.18b}$$

En el caso de que sí existan pérdidas, es posible calcularlas añadiendo dicho término:

$$|S_{11}|^2 + |S_{21}|^2 + \text{Loss}_1 = 1 \tag{2.19a}$$
$$|S_{22}|^2 + |S_{12}|^2 + \text{Loss}_2 = 1 \tag{2.19b}$$

Por otro lado, la fase $\phi(\omega)$ de la red, necesaria para calcular el retardo de grupo negativo (Ecuación 1.2), se corresponde con la fase del coeficiente de transmisión:

$$\phi(\omega) = \arg(S_{21}) \quad \text{(rad)}. \tag{2.20}$$

2.1.3.2. Matriz de transmisión ABCD

Mientras que la matriz de Scattering permite describir el sistema en función de las relaciones entre sus puertos, la matriz de transmisión ABCD facilita la conexión en cascada de redes de dos puertos. Esto resulta altamente interesante a la hora de realizar un análisis de una red compuesta de varios componentes en cascada. La relación de la

matriz ABCD en una red de dos puertos es:

$$\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} V_2 \\ I_2 \end{bmatrix} \tag{2.21}$$

Ante una conexión en serie de dos o más redes de dos puertos Figura 2.4, es posible calcular la matriz ABCD equivalente como:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix} \begin{bmatrix} A_2 & B_2 \\ C_2 & D_2 \end{bmatrix} \begin{bmatrix} A_N & B_N \\ C_N & D_N \end{bmatrix} \tag{2.22}$$

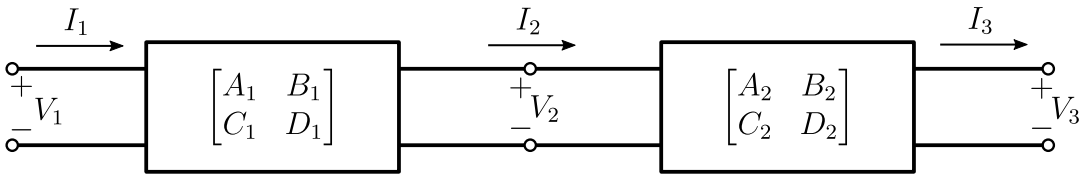


Figura 2.4: Representación general de una red de dos puertos

En la literatura podemos encontrar equivalentes de la matriz ABCD para circuitos de dos puertos comunes [45]. Sin embargo, ya que en este TFM la red de microondas bajo análisis está compuesta únicamente por resistencias y tramos de líneas de transmisión (como se verá más adelante en el Capítulo 3), en la Tabla 2.1 se expresan estos dos únicamente:

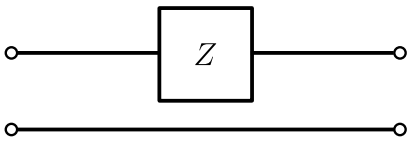
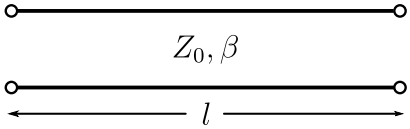
Circuito	Parámetros ABCD
	$A = 1 \quad B = Z$ $C = 0 \quad D = 1$
	$A = \cos(\beta\ell) \quad B = jZ_0 \sin(\beta\ell)$ $C = jY_0 \sin(\beta\ell) \quad D = \cos(\beta\ell)$

Tabla 2.1: Parámetros ABCD equivalentes

donde $\theta = \beta\ell$ (rad) es la longitud eléctrica.

Por otro lado, si la red bajo análisis se compone de dos ramas en paralelo, siguiendo la segunda regla de la descomposición de diagramas de flujo, el equivalente es la suma [45].

Siguiendo esta regla, para realizar el análisis de este TFM, obtendremos el equivalente de las ramas en paralelo mediante la matriz de admitancia:

$$\begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} = \begin{bmatrix} Y_{11_1} & Y_{12_1} \\ Y_{21_1} & Y_{22_1} \end{bmatrix} + \begin{bmatrix} Y_{11_2} & Y_{12_2} \\ Y_{21_2} & Y_{22_2} \end{bmatrix} \tag{2.23}$$

Ya solo resta definir cómo se relacionan entre sí los parámetros de todas las matrices repasadas en este capítulo. Para ello y a modo de resumen, se adjunta la Tabla A.1 [45] en el Anexo A.

3. Topología bajo estudio y objetivos

El objetivo de este TFM, como bien indica su título, es realizar un estudio de estructuras con retardo de grupo negativo en tecnología microstrip. La topología inicialmente propuesta se muestra en la Figura 3.1. A dicha topología nos referiremos como Modelo 1.

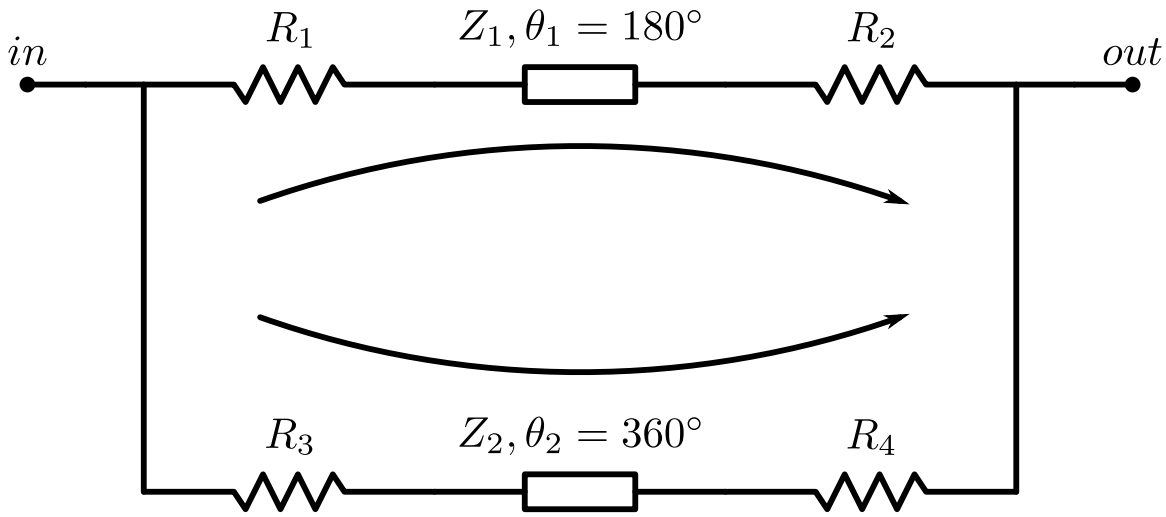


Figura 3.1: Topología propuesta: Modelo 1

En un principio, los resultados que se esperaban obtener eran similares o mejores a los expuestos en el artículo *Signal-Interference-Based Structure with Negative Group Delay Properties* (2018) [16]. La diferencia entre la topología de la Figura 3.1 y la publicada en dicho artículo reside en las longitudes eléctricas de los tramos de línea superior (TL_1) e inferior (TL_2). En ambos casos nos encontramos ante un dispositivo interferencial ya que la señal llega desfasada 180°, sin embargo las longitudes de la topología del artículo son $\theta_1 = 90°$ y $\theta_2 = 270°$. Esta diferencia supone que en la topología propuesta en este TFM existan tres ceros de transmisión. Por lo tanto, el objetivo pasaba por ajustar las pérdidas introducidas por las resistencias para así suavizar la respuesta y conseguir una banda con NGD muy ancha. Por desgracia, no es posible para valores de NGD interesantes y atenuaciones plausibles.

Sin embargo, el sistema tiene varios grados de libertad que merece la pena estudiar ya que para configuraciones concretas es posible obtener dispositivos con retardo de grupo negativo de una, dos o tres bandas.

El análisis de una topología pasiva como la del Modelo 1 es relativamente sencillo gracias a la teoría de análisis de redes de microondas vista en el Capítulo 2. Sin embargo, tal y como se expuso en la Ecuación 1.2, el retardo de grupo negativo se obtiene a partir de la derivada de la fase, y puesto que contamos con tres ceros de transmisión donde sólo el central se mantiene constante en frecuencia, es difícil simplificar la expresión resultante porque no se conoce a priori la frecuencia de las bandas ya que depende de los valores de impedancias y resistencias de la topología. Esta característica lleva a que obtener expresiones que caractericen completamente el dispositivo en cuanto a retardo de grupo negativo no sea factible analíticamente, haciendo muy difícil obtener ecuaciones o curvas de diseño cerradas.

Los métodos tradicionales para realizar estudios paramétricos se basan en fijar una de las variables y modificar el resto. Por desgracia, para esta topología este tipo de estudios deja muchísimas combinaciones potenciales fuera, por lo que si se pretende tener un control más exhaustivo de la topología es necesario llevar a cabo un análisis paramétrico por fuerza bruta. Gracias a la potencia computacional de las máquinas actuales es posible aplicar la teoría de circuitos y de análisis de redes de microondas para obtener, junto con un pequeño procesado de señal, datos que caractericen el sistema.

En los siguientes capítulos veremos más en detalle cómo se ha desarrollado este análisis y los resultados y conclusiones obtenidos.

4. Desarrollo

En este capítulo se presenta el desarrollo del estudio así como las herramientas utilizadas y algunas de sus diferencias más características.

4.1. Estudio Paramétrico

Tal y como ya se ha comentado anteriormente, el estudio paramétrico de la topología propuesta se ha realizado por fuerza bruta mediante herramientas de computación como Python y MATLAB ® y otras más específicas del diseño de sistemas como ADS y Sonnet.

4.1.1. Cálculo de la respuesta de la topología

Para obtener la respuesta ideal del circuito podemos optar por herramientas específicas como ADS, que permiten generar el esquema eléctrico de la Figura 3.1 con los valores reales para cada parámetro. En la Figura 4.1 se muestra el circuito creado en ADS. En el Anexo B se detalla el uso de este software.

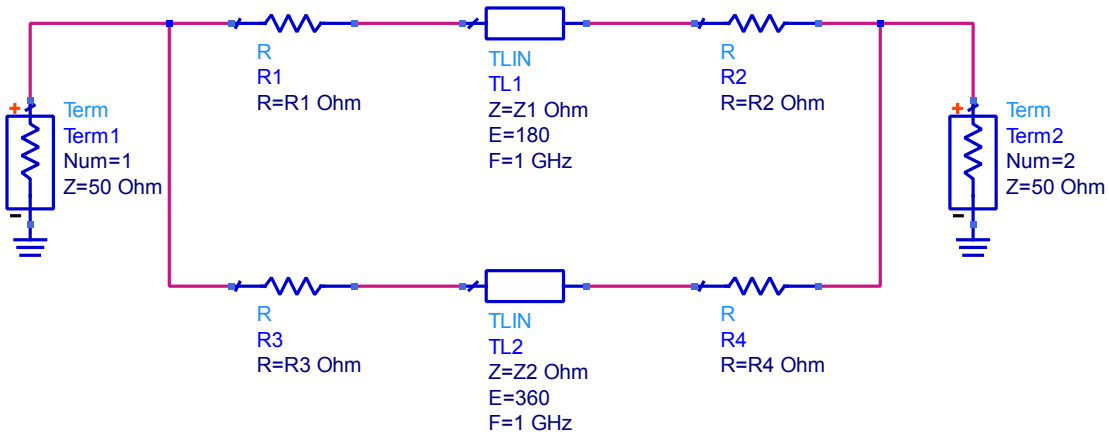


Figura 4.1: Circuito simulado en (ADS) (Modelo 1)

Como puede observase, a la entrada y la salida del dispositivo se han conectado dos **Term** (Terminales) que simbolizan la impedancia de los puertos ($Z = 50\ \Omega$ en todos los casos de este TFM).

Por otro lado, cabe destacar que el dispositivo bajo análisis tiene 6 grados de libertad, es decir tiene 6 variables: 4 resistencias y las 2 impedancias características de las líneas de transmisión. Dichos valores se referencian mediante el objeto **VAR** que permite, entre otras, su variación en tiempo real mediante la opción **Tunning**. Por ejemplo para los valores de Resistencias e Impedancias de la Tabla 4.1, se obtiene la respuesta de la Figura 4.2, donde se muestran el retardo de grupo (en segundos) y los parámetros $S_{2,1}$, $S_{1,1}$ y $S_{2,2}$ (en decibelios), junto con marcadores en las dos frecuencias de interés y la central a 1 GHz.

$R_1(\Omega)$	$Z_1(\Omega)$	$R_2(\Omega)$	$R_3(\Omega)$	$Z_2(\Omega)$	$R_4(\Omega)$
5	55	5	5	40	100

Tabla 4.1: Valores para el Modelo 1

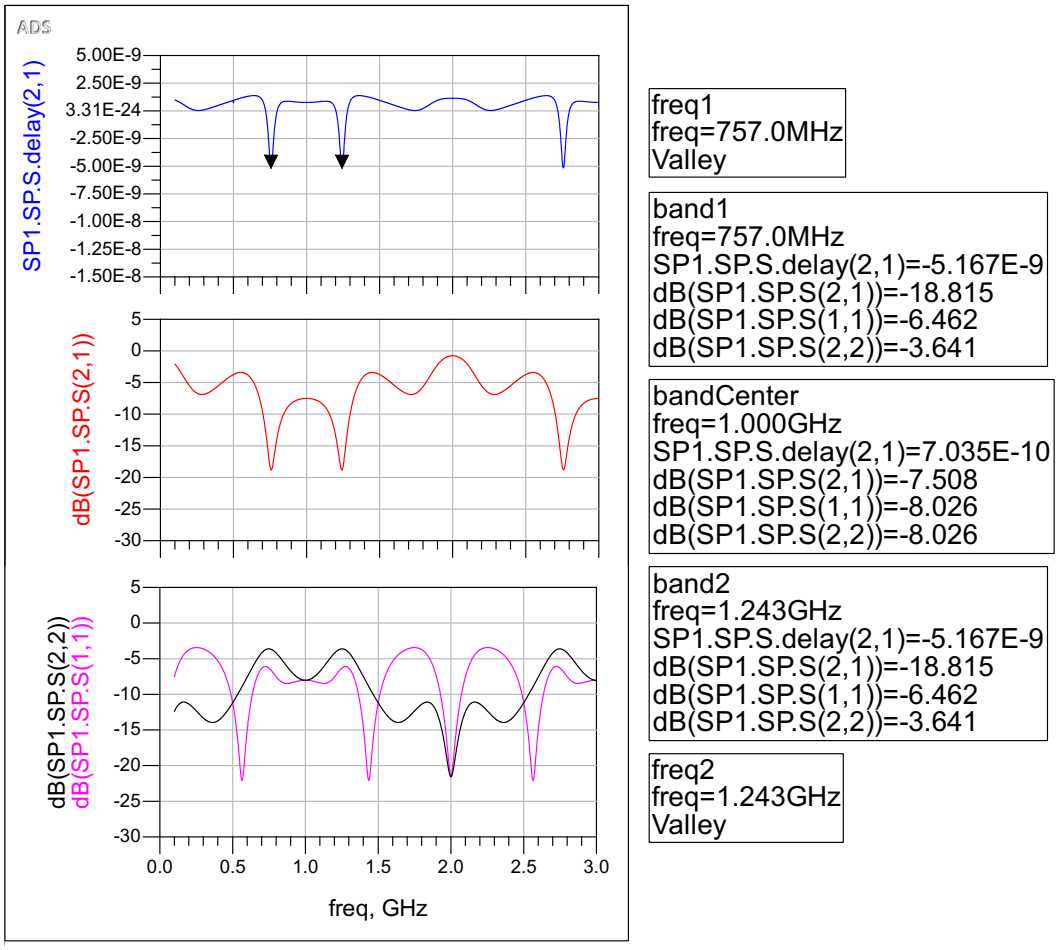
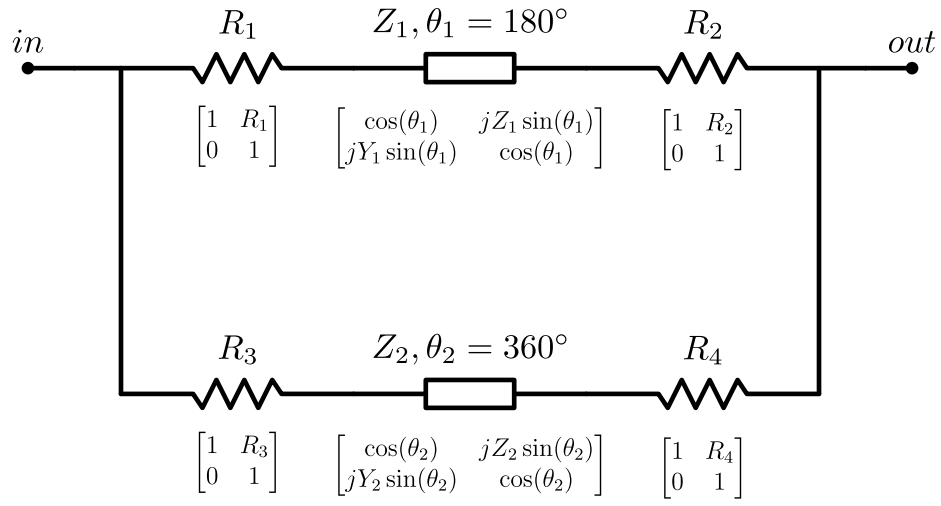
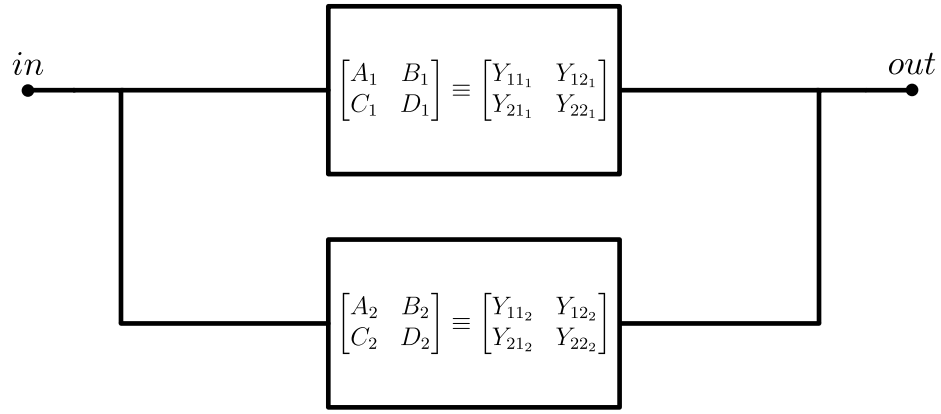


Figura 4.2: Respuesta del Modelo 1 con la configuración de la Tabla 4.1



(a) Modelo 1 en matrices $ABCD$ para cada componente



(b) Modelo 1 analizado como dos ramas en paralelo

Figura 4.3: Análisis de redes de microondas aplicado al Modelo 1

Otra opción para obtener la respuesta del circuito ideal es realizar el cómputo en Matlab ®. Para ello es necesario aplicar el análisis de redes de microondas del Capítulo 2. En la Figura 4.3a se muestra la matriz $ABCD$ para cada uno de los componentes del circuito, de dónde podemos obtener el equivalente para cada rama en serie:

$$\begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix} = \begin{bmatrix} 1 & R_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_1) & jZ_1 \sin(\theta_1) \\ jY_1 \sin(\theta_1) & \cos(\theta_1) \end{bmatrix} \begin{bmatrix} 1 & R_2 \\ 0 & 1 \end{bmatrix} \tag{4.1a}$$

$$\begin{bmatrix} A_2 & B_2 \\ C_2 & D_2 \end{bmatrix} = \begin{bmatrix} 1 & R_3 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_2) & jZ_2 \sin(\theta_2) \\ jY_2 \sin(\theta_2) & \cos(\theta_2) \end{bmatrix} \begin{bmatrix} 1 & R_4 \\ 0 & 1 \end{bmatrix} \tag{4.1b}$$

Obteniendo así una red que puede representarse mediante la Figura 4.3b y es posible convertir dichas matrices de transmisión a matrices de admitancias mediante la

Tabla A.1 del Anexo A. De esta forma, tal y como ya se ha comentado, podríamos obtener la matriz de admitancia equivalente mediante la suma de ambas matrices de admitancia de ambas ramas:

$$[Y] = [Y_1] + [Y_2] \tag{4.2}$$

Y, mediante la Tabla A.1 convertirla a la matriz de Scattering, obteniendo así la matriz de dispersión con la que representar la respuesta de la red.

Esta matriz de dispersión $[S]$ sería la respuesta del circuito para unos valores de resistencias e impedancias determinados y una frecuencia concreta. Sin embargo, es posible trabajar con un vector de frecuencias normalizado f_n y, posteriormente, desnormalizar para obtener los valores en función de la frecuencia real:

$$\boldsymbol{f} = \boldsymbol{f_n} f_0 \tag{4.3a}$$

$$\boldsymbol{\theta} = \theta \boldsymbol{f_n} \tag{4.3b}$$

Por lo tanto, todos los parámetros de dispersión obtenidos están en función de la frecuencia normalizada, por lo que para representarlos hemos de tenerlo en cuenta. Sólo restaría calcular el retardo de grupo, que como se expresó en la Ecuación 1.2 se calcula mediante la derivada de la fase del coeficiente de transmisión:

$$\tau_g(\omega) = -\frac{\partial \arg(S_{21}(\omega))}{\partial \omega}, \tag{4.4}$$

Al tratarse de una derivada, es importante *desenrollar* la fase del coeficiente de transmisión para evitar saltos en dos puntos consecutivos. Por lo que para programar dicha función en MATLAB ® resultaría la siguiente línea de código:

```
group_delay = -(diff(unwrap(angle(s21),pi))./diff(f))/(2*pi);
```

En el Anexo D se incluyen las funciones pertinentes para realizar la simulación así como el script de prueba para computar casos concretos. El resultado del Modelo 1 con los parámetros de la Tabla 4.1 se muestra en la Figura 4.4.

4.1.2. Análisis de la respuesta de la topología

Aunque la simulación de la topología es útil para obtener su respuesta, los métodos mencionados en el apartado anterior son válidos para unos valores de resistencias e impedancias determinados. Sin embargo, de cara al diseño de dispositivos para sistemas de comunicación específicos, es necesario caracterizar la topología de modo que ante unos requerimientos de **atenuación**, **retardo de grupo negativo** o **ancho de banda**, sea posible elegir los valores adecuados para obtener dicha respuesta. Tal y como se comentó en el Capítulo 3, para la topología propuesta no es posible encontrar expresiones aproximadas a partir del análisis matemático del sistema y por ello se opta por un análisis paramétrico por fuerza bruta.

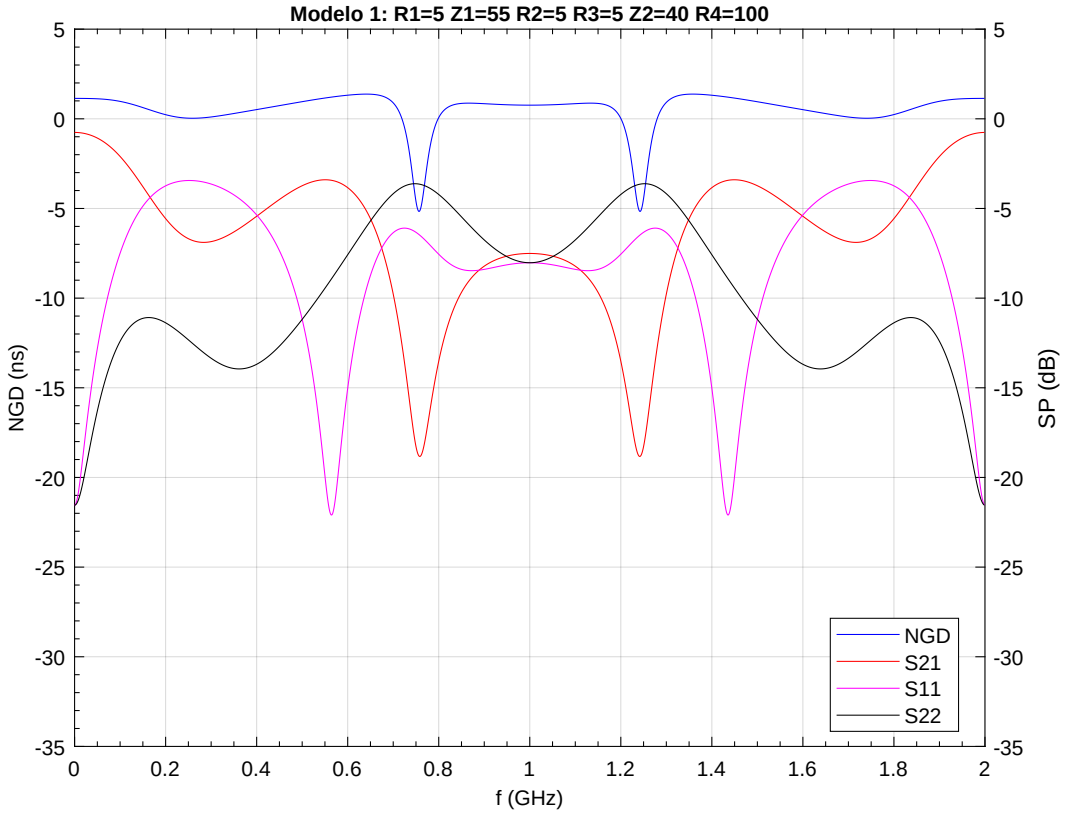


Figura 4.4: Respuesta del Modelo 1 con la configuración de la Tabla 4.1 simulada en Matlab ®

Para llevar a cabo este análisis por fuerza bruta en un inicio se utilizó un componente de ADS llamado *Parameter Sweep*. Este componente nos permite configurar la simulación de modo que se repita para diversos valores de una misma variable. Por desgracia, aunque preciso y sencillo de utilizar, este método es lento ya que genera datos que no necesitamos y los almacena en memoria.

Siguiendo este mismo principio, se construye un sistema de simulación automática para diversos valores de las variables del circuito. Las resistencias pueden oscilar entre 0 Ω (circuito cerrado) y 100 Ω, que son valores plausibles para su fabricación y que no supondrían un exceso de atenuación. Mientras que las impedancias varían entre 25 y 125 Ω para evitar efectos de radiación debidos a las dimensiones de la línea conductora (cota inferior) y no obtener anchuras demasiado estrechas que no sea posible fabricar (cota superior). En ambos casos el incremento es de 5 Ω.

Por sencillez en el modo de programación y posibilidad de escalado, este sistema se implementa en Python. Al tratarse de un lenguaje de programación *open-source* es sencillo instalar el compilador en otras máquina sin depender de licencias. Esto ha

posibilitado que gran parte de los cálculos se realicen en diversas máquinas, no sólo físicas, sino también virtuales mediante entornos como Amazon Web Services o Google Cloud Platform. Estos servicios permiten crear instancias de máquinas virtuales con sistemas operativos Linux donde ejecutar remotamente la simulación y, posteriormente, descargar los datos generados.

La traducción de las funciones generadas en MATLAB ® para la simulación de la topología a Python es relativamente sencilla, teniendo que adaptar algunas funciones y la sintaxis. La diferencia principal reside en el script de ejecución. Para realizar el análisis por fuerza bruta es necesario simular el modelo para cada caso, siendo un caso una combinación concreta de los valores de las variables (resistencias e impedancias). Teniendo en cuenta el número de variables y el rango de éstas, el número de casos a computar sería:

Variable	**Rango**	**Incremento**	**N**
Z_1 y Z_2	$[25, 125]$	5	21
R_1, R_2, R_3 y R_4	$[0, 100]$	5	21
f_n	$[0, 2]$	0.001	2001

Tabla 4.2: Variables y posibles valores

Para el Modelo 1 existen $21^6 = 85.766.121$ casos diferentes. No se tiene en cuenta el vector de frecuencias ya que el mayor o menor número de puntos en este aumenta o disminuye la precisión de los cálculos, pero no influye en el número de combinaciones posibles. Por supuesto que todos estos casos pueden simularse mediante bucles anidados, sin embargo, existen formas más eficientes.

Debido a las características de los cálculos a realizar, se opta por un procesado en paralelo gracias a los procesadores multinúcleo actuales. Sin adentrarnos en detalle en este tema, este tipo de ejecución nos permite simular tantos casos como núcleos lógicos tenga el procesador al mismo tiempo. Implementar esta característica es relativamente sencillo en Python gracias a la librería *multiprocessing*:

```
M = list(xRange) # variable 1
N = list(xRange) # variable 2
contador = list(it.product(M, N)) # combinación de ambas variables
# Creación del Pool de procesos a partir del número de núcleos de la cpu
p = multiprocessing.Pool(multiprocessing.cpu_count())
# Aplica la función calcular en paralelo para todos los valores de contador
p.map(calcular, contador)
# Cierra y termina el multiprocesado
p.close()
p.join()
```

Puesto que tras la simulación de cada caso es necesario almacenar en un archivo los resultados, tampoco es conveniente generar 21^6 archivos. El sistema de multiprocesado de Python permite almacenar el resultado en una variable cargada en memoria, sin embargo esta variable también sería demasiado pesada para casi cualquier máquina.

Por lo tanto, la solución más eficiente pasa por fraccionar la simulación y almacenar conjuntos de simulaciones. El sistema reparte a cada núcleo el valor de dos variables, las que se corresponderían con las capas más externas de los bucles anidados, y cada núcleo ejecuta, ahora sí, cuatro bucles anidados que producen una matriz de datos que se almacena en un archivo. En la Figura 4.5 se muestra este mismo concepto en forma de esquema visual.

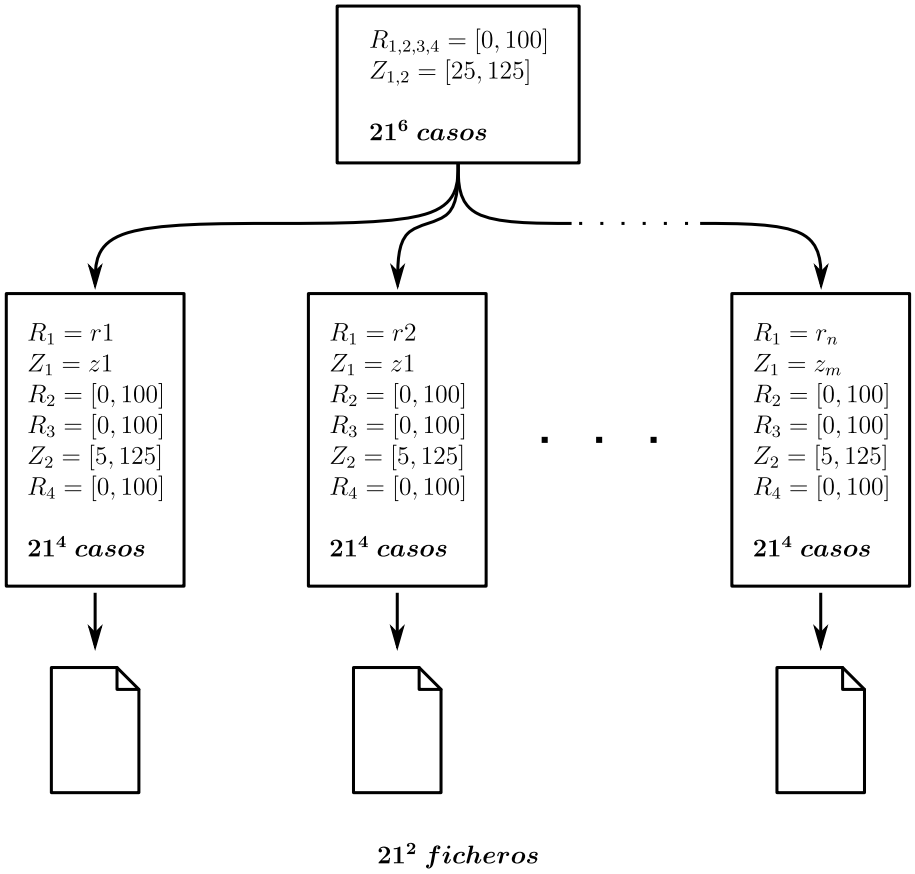


Figura 4.5: Esquema del multiprocesado por casos del Modelo 1

En el Anexo E se adjuntan todas las funciones y scripts necesarios para realizar la simulación.

La simulación de la respuesta de los casos es relativamente rápida gracias al cálculo matricial del análisis de la red, que evita que se tengan que utilizar bucles para evaluar el sistema con cada valor de frecuencia. Sin embargo, la respuesta para cada caso, en forma de vector en función de la frecuencia, no nos permite realizar un análisis paramétrico del sistema. Para ello es necesario introducir un pequeño procesado de señal para extraer los valores de interés. En este sentido, se consideran de interés los parámetros de la Tabla 4.3.

Para obtener estos datos es necesario necesario encontrar los mínimos de la señal

Parámetro	Tipo	Descripción
$\boldsymbol{NGD}$	3x1 (float)	Retardo de grupo negativo para cada banda, incluida la central, si lo hubiera.
$\boldsymbol{f(NGD)}$	3x1 (int)	Índice del vector frecuencia donde se encuentran las bandas.
$\boldsymbol{S_{21}}$	3x1 (float)	Coeficiente de transmisión para cada banda.
$\boldsymbol{S_{11}}$	3x1 (float)	Coeficiente de reflexión de entrada para cada banda.
$\boldsymbol{S_{22}}$	3x1 (float)	Coeficiente de reflexión de salida para cada banda.
$\boldsymbol{f(BW)}$	3x2 (int)	Índice del vector frecuencia donde empieza y acaba el ancho de banda con ngd, medido éste como aquel ancho de banda donde el retardo de grupo es negativo.
$\boldsymbol{f(BW_{1dB})}$	3x2 (int)	Índice del vector frecuencia donde empieza y acaba el ancho de banda con ngd, medido éste como aquel ancho de banda donde la diferencia del parámetro S_{21} es de 1 dB sobre su mínimo.
$\boldsymbol{f(BW_{3dB})}$	3x2 (int)	Índice del vector frecuencia donde empieza y acaba el ancho de banda con ngd, medido éste como aquel ancho de banda donde la diferencia del parámetro S_{21} es de 3 dB sobre su mínimo.

Tabla 4.3: Parámetros de interés para la caracterización paramétrica

y discernir si son los adecuados ya que en el retardo de grupo puede haber diversos mínimos. Es posible acotar la posición de los mínimos, es decir de las bandas de interés. Mientras que la banda central se encuentra siempre para la frecuencia central (1 GHz en este caso); las bandas laterales son siempre simétricas para el Modelo 1 y no se corresponden exactamente con el cero de transmisión del parámetro S_{21}. Por ello matemáticamente no es posible calcular su frecuencia, sino que tenemos que recurrir a la búsqueda de dicho mínimo. Para optimizar la búsqueda se aplican las siguientes condiciones:

- La banda de frecuencia inferior (Banda 1) se encontrará entre 0 y 999 MHz
- El retardo de grupo debe ser negativo
- El ancho de dicho pico mínimo debe ser superior o igual a 2

De estas condiciones, merece la pena destacar el motivo de la última: las singularidades matemáticas. Al tratarse de un cálculo numérico con divisiones, derivadas y números complejos del que finalmente se obtiene un valor real, existen puntos inconsistentes que arrojan valores infinitos que se traducen en picos mínimos o máximos

de retardo de grupo entre dos muestras consecutivas. Al someter a todos los mínimos locales encontrados a dicha condición, descartamos aquellos que se deban a singularidades matemáticas. En la Figura 4.6 se muestran los dos mínimos encontrados con su correspondiente ancho (línea roja horizontal) y la altura de dicho mínimo (línea negra vertical). Cabe destacar que el ancho del mínimo encontrado no se corresponde con el ancho de banda que se menciona en la Tabla 4.3, sino que es el ancho matemático de dicho mínimo local.

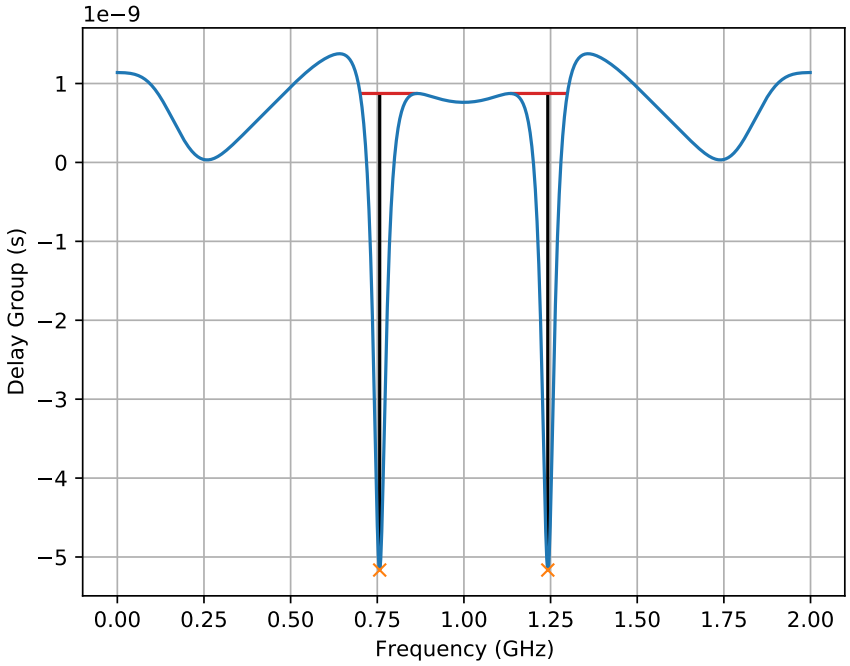


Figura 4.6: Búsqueda de mínimos de interés

Los datos generados se almacenan en un archivo de extensión *.mat* en forma de matriz donde constan los valores de cada variable (resistencias e impedancias) y de los parámetros de interés de la Tabla 4.3. Cabe resaltar que puesto que las bandas laterales son simétricas sólo se procesa y almacena la información relativa a la inferior.

El tiempo de cómputo para un equipo con un procesador Intel Core i7-10710U con 6 núcleos físicos a 1,10 GHz, 16 GB de memoria RAM y la versión 3.8.1 de Python es de 10,5 horas al 100% de su capacidad de cómputo.

Posteriormente se realiza un pre-procesado en MATLAB ® donde se cargan los datos y se convierten a un formato propio denominado DataStore que permite cargarlos fraccionadamente en memoria. Además se convierte la matriz en una tabla, lo que facilita el acceso a los datos mediante el identificador de cada columna (D.4).

Llegados a este punto, la tabla de datos generados tiene unas dimensiones de 85.776.121

filas y 28 columnas (variables y parámetros de interés). Sin embargo, no todos los casos dan respuestas de interés: algunas combinaciones generan circuitos con retardos de grupo positivos, con atenuaciones muy altas o con NGD cercanos a cero. Por lo tanto, se analizan los datos para obtener tres DataStore más concretos (Código D.5):

- Doble Banda: Únicamente las bandas laterales tienen NGD, inferior a -1 ns y con atenuación inferior a 20 dB.
- Banda Central: Únicamente la banda central cumple dichas condiciones.
- Triple Banda: Las tres bandas cumplen dichas condiciones.

El análisis paramétrico se realiza a partir de dichos DataStore.

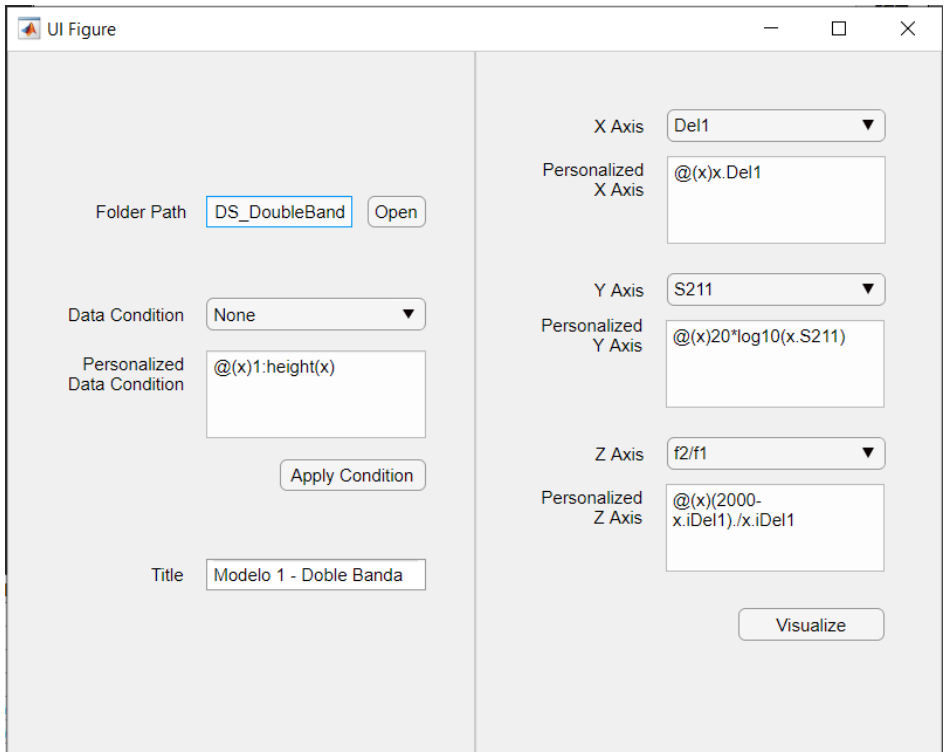
4.1.3. Análisis paramétrico

El análisis de datos con dependencia de múltiples variables puede ser complejo e incluso inabordable ante la cantidad de datos del modelo bajo estudio. Para arrojar algo de luz sobre las variables de interés y el comportamiento de la red, se ha implementado una interfaz gráfica en MATLAB ® (Figura 4.7a) que carga los DataStore y permite seleccionar qué variables o parámetros queremos representar unos frente a otros como una nube de puntos.

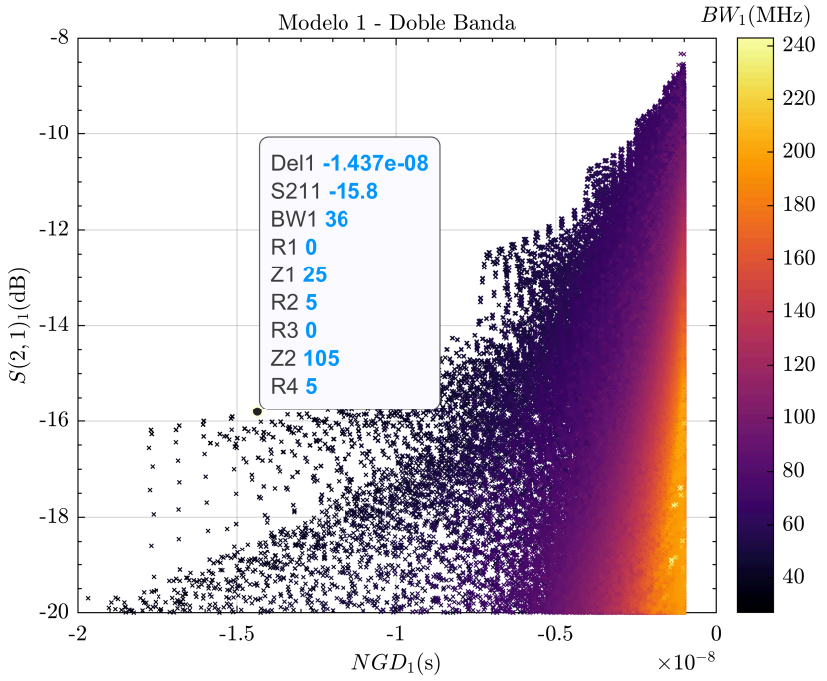
Es posible aplicar condiciones personalizadas a los datos o representar en función de la relación entre varias variables gracias a la flexibilidad que otorga trabajar con datos estructurados en tablas en MATLAB ®. También es posible representar en 2 o 3 dimensiones, aunque la tercera dimensión es más interesante en modo gradiente de color. Por otro lado, se ha implementado una funcionalidad que añade a cada *Data Point* (desplegable de MATLAB ® que da información sobre un punto de la gráfica) los valores de las resistencias e impedancias, haciendo esta forma muy sencillo localizar a qué caso corresponde la respuesta de interés (Figura 4.7b).

La aplicación se diseña y programa en AppDesigner (MATLAB ®) y el código completo se encuentra en el Apartado D.6 del Anexo D.

En ocasiones también puede resultar interesante analizar la distribución de los valores que toman las variables con mayor o menor frecuencia para condiciones determinadas. En este sentido se han implementado una función para calcular el histograma (en paralelo) y representarlo (Figura 4.8). El código de la función y el script se adjunta en los Apartados D.7 y D.8.



(a) Entorno gráfico de la aplicación



(b) Ejemplo de representación de análisis del Modelo 1

Figura 4.7: Aplicación desarrollada para analizar paramétricamente el sistema

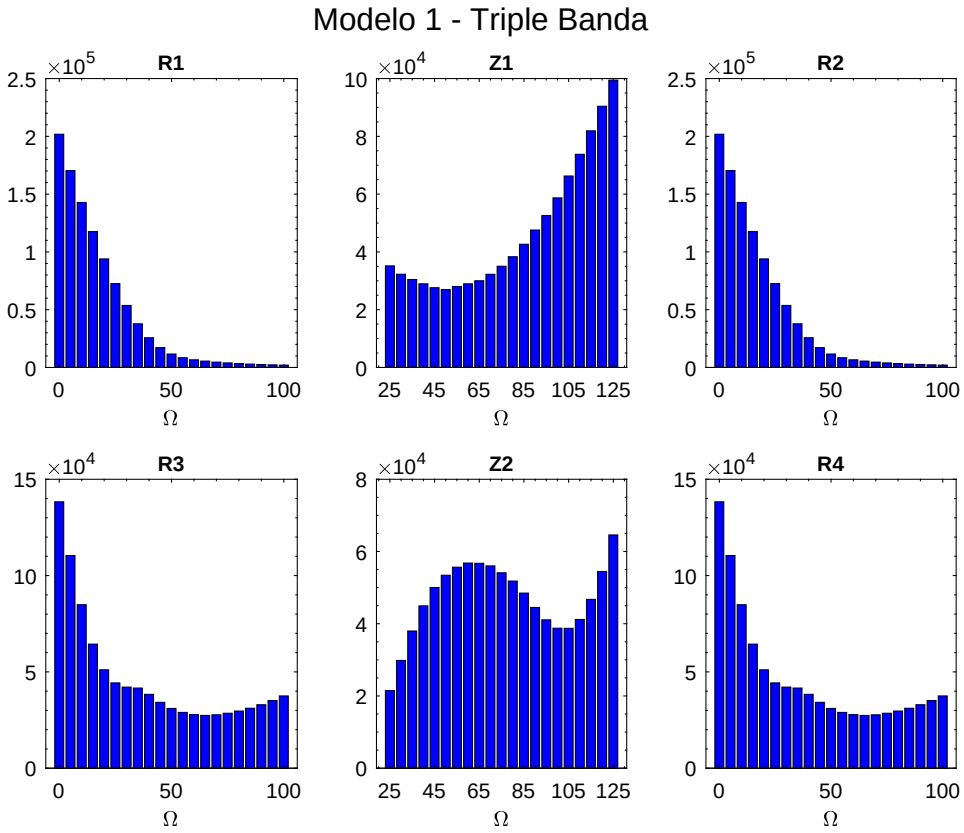
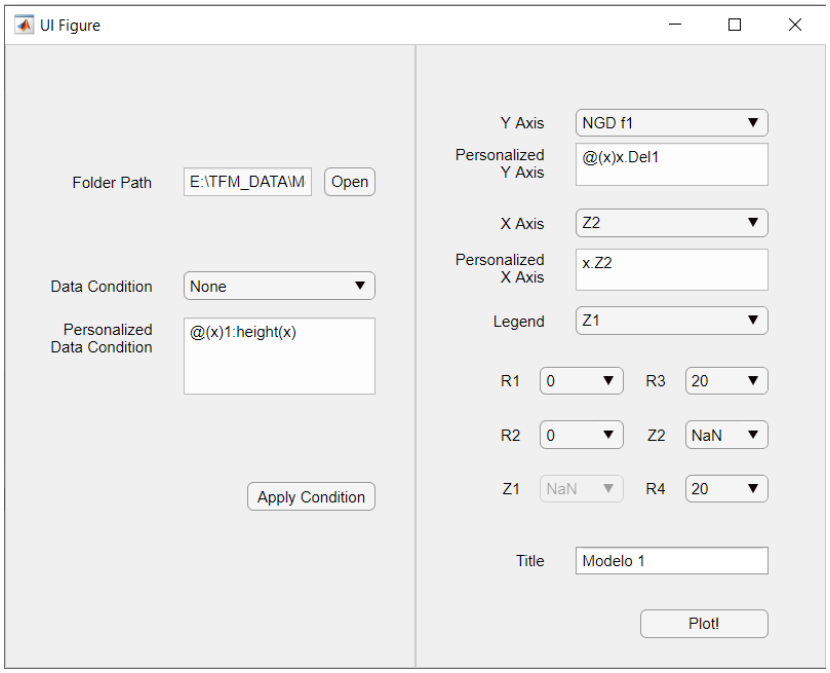


Figura 4.8: Histogramas de las variables para el Modelo 1 - Doble Banda

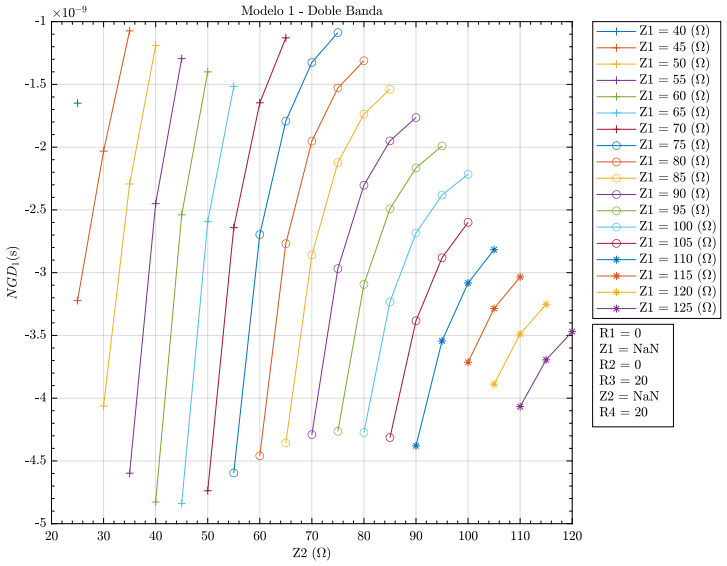
4.1.4. Curvas de diseño

El objetivo final del análisis paramétrico es la obtención de curvas de diseño que nos permitan, en base a unas especificaciones concretas, conocer con rapidez el valor que debemos utilizar para cada variable. En este sentido se ha implementado un entorno gráfico que permite generar gráficas de un parámetro frente a dos variables, una de ellas a modo de leyenda.

La interfaz es relativamente similar a la anterior y se puede ver, junto con un ejemplo de curva de diseño, en la Figura 4.9. El código completo se incluye en el Apartado D.9 del Anexo D.



(a) Entorno gráfico de la aplicación



(b) Ejemplo de una curva de diseño del Modelo 1

Figura 4.9: Aplicación desarrollada para generar curvas de diseño

4.2. Diseño de Prototipos

En apartado Líneas Microstrip del capítulo 2 se ha ilustrado la relación entre los parámetros eléctricos y físicos de una línea microstrip. En este sentido, es posible realizar un análisis, para calcular los parámetros eléctricos de unas dimensiones conocidas, o sintetizar dichas dimensiones a partir de los parámetros eléctricos deseados. En este apartado se expone cómo se ha llevado a cabo dicha síntesis, los cálculos adicionales para adecuar las dimensiones al circuito y la posterior simulación tanto en ADS como en un simulador de onda completa como es Sonnet.

4.2.1. Síntesis de los parámetros eléctricos

A partir de las ecuaciones 2.6, 2.7, 2.8, 2.9 y 2.10 es posible analizar o sintetizar el dispositivo. Por suerte, multitud de software e incluso páginas web permiten realizar estos cálculos de forma automática. En este TFM se ha utilizado la herramienta adicional de ADS LineCalc. En la Figura 4.10 se muestra la interfaz de esta aplicación.

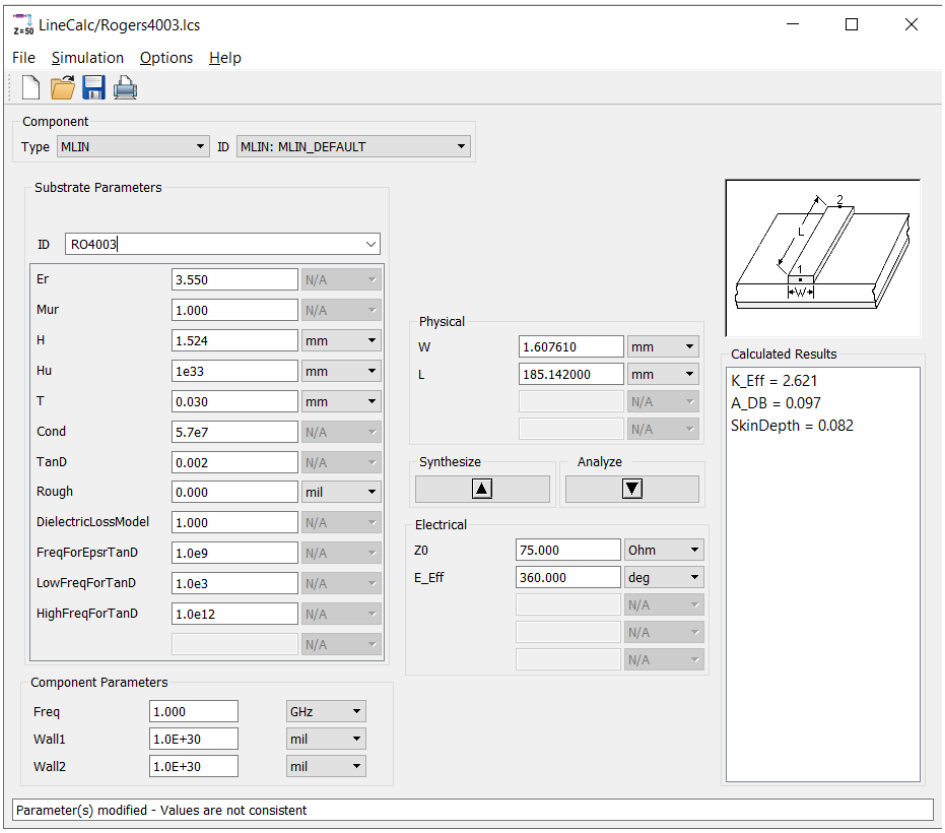


Figura 4.10: Herramienta LineCalc de ADS

En la Tabla 4.4 se describen los parámetros de relevancia y se especifican para dos substratos comerciales: Rogers 4003 y TLX-8.

ID	Unidad	Ro4003	TLX-8	Descripción
Er	Adim.	3,550	2,550	Permitividad efectiva relativa ($\epsilon_{r,eff}$) del dieléctrico.
H	mm	1,524	0,790	Espesor del substrato.
Hu	mm	1e33		Posición de la tapa. Si no la hay se configura con un valor muy grande.
T	mm	0,030		Espesor del conductor (cobre).
Cond	Adim.	5,7e7		Conductividad del conductor (cobre).
TanD	Adim.	0,0027	0,0017	Tangente de pérdidas.

Tabla 4.4: Parámetros de configuración de LineCalc

Por otro lado, el diseño físico del Modelo 1 precisa de curvar o doblar alguna de las líneas de transmisión para poder unirlas en el mismo nodo. Concretamente, ya que la línea de transmisión inferior tiene una longitud eléctrica de 360° frente a los 180° de la superior, será la de mayor longitud. Esto se traduce en un diseño con las características del esquema de la Figura 4.11.

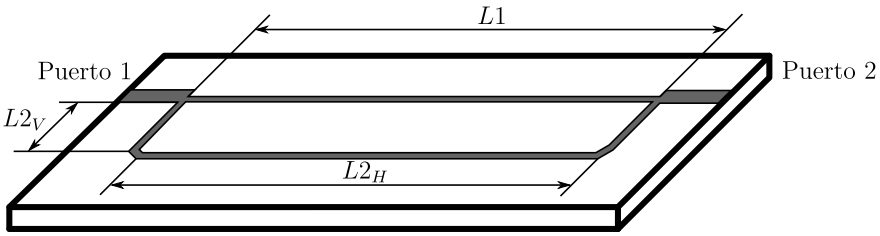


Figura 4.11: Esquema genérico de implementación del Modelo 1

Por lo que habrá que dividir la línea de transmisión inferior en tres secciones unidas mediante un ángulo. Para calcular estos tramos hay que tener en cuenta el espacio que ocupan las resistencias superiores, si las hubiera. Las resistencias comerciales son del tipo SMD 0603, por lo que precisan de 0,5 mm entre líneas, por lo que, como se muestra en el código 4.1, es necesario indicar también el número de las mismas.

Código 4.1: Script para el cálculo de los tramos de la línea inferior

```
% Width and Length for TL1 and TL2
W1 = 0.5;
L1 = 95.4;
W2 = 1.6;
L2 = 185.2;
% Number of resistences
num_r = 2;
```

```
% Calc.
r_space = 0.5;
L2_h = L1 + num_r*r_space;
L2_v = (L2 - L2_h - 2*W2)/2;
```

4.2.2. Simulación Microstrip (ADS)

Al igual que en el apartado 4.1.1 Cálculo de la respuesta de la topología, es posible simular la respuesta del dispositivo mediante su equivalente con líneas de transmisión microstrip en ADS. En la Figura 4.12 se muestra dicho circuito, el cual es muy parecido a su equivalente ideal, pero con la diferencia de que es necesario añadir las conexiones en 'T' que supone conectar dos líneas de transmisión con el puerto de entrada o salida en un mismo nodo. Puesto que estos puntos son críticos en cuanto a efectos parásitos y comportamientos indeseados, es importante tenerlos en cuenta a la hora de simular.

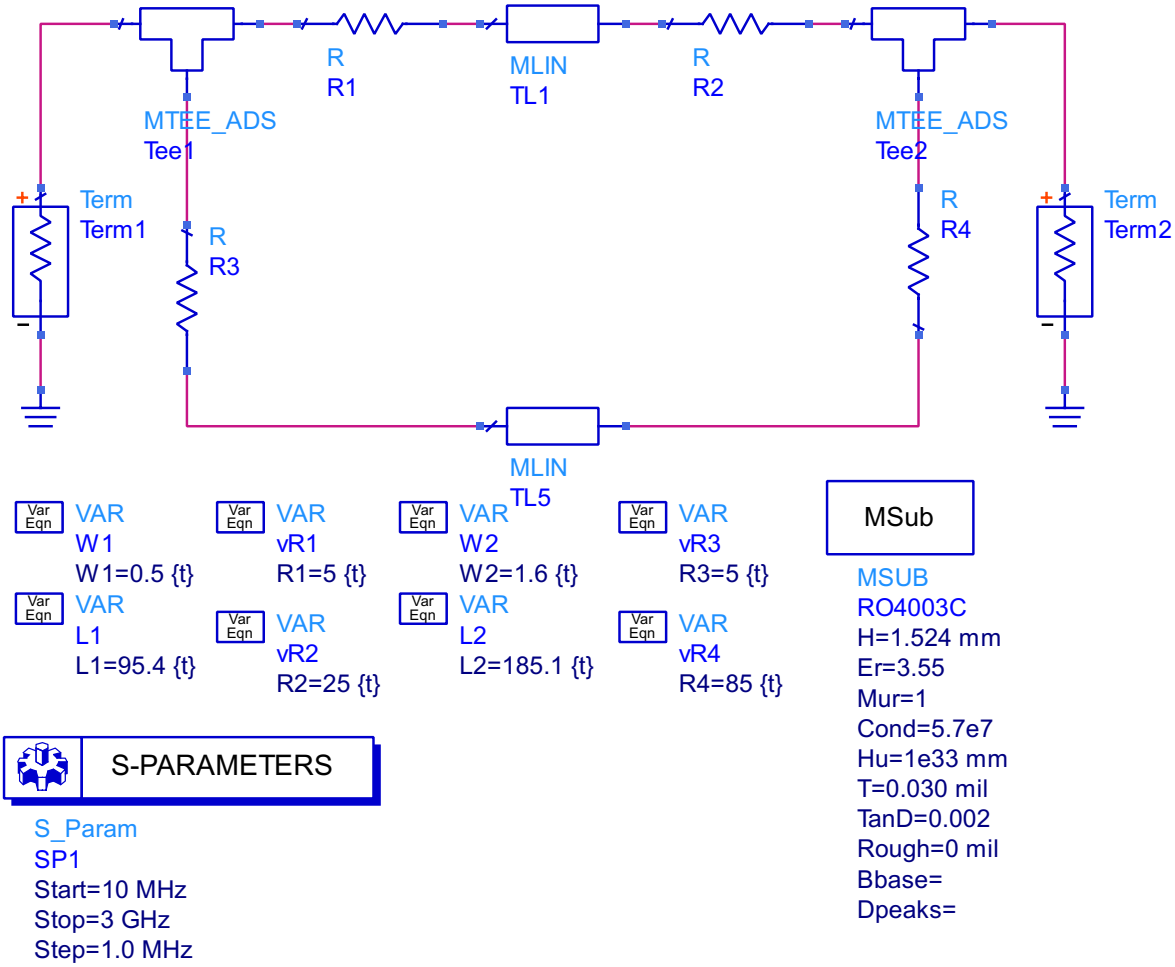


Figura 4.12: Circuito simulado en ADS con líneas microstrip (Modelo 1)

Para hacer mostrar el resultado de la simulación, se sintetizan los valores de la Tabla 4.1, obteniéndose las medidas de longitud y ancho para cada línea de la Tabla 4.5.

Línea de Transmisión	$Z_0(\Omega)$	$\theta_0(\Omega)$	W(mm)	L(mm)
Superior (TL1)	55	180°	$2,87$	$90,53$
Inferior (TL2)	40	360°	$4,75$	$177,17$

Tabla 4.5: Dimensiones físicas para las líneas de transmisión del Modelo 1

Tal y como se ha comentado, las conexiones en 'T' son un punto crítico y un mal diseño puede ocasionar grandes variaciones en la respuesta del circuito. En este sentido, la documentación de ADS indica cómo se configuran los puertos de la 'T' en función de su anchura. En la Figura 4.13 se muestra la nomenclatura para cada puerto y su relación con las anchuras del resto de puertos.

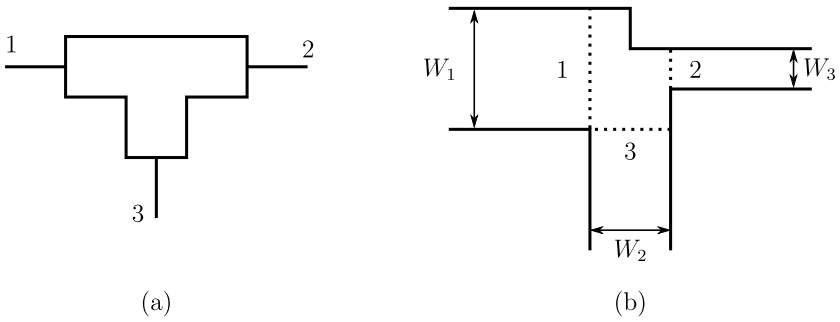


Figura 4.13: Diagrama de la relación entre los anchos del conector en 'T' y sus puertos. (a) Símbolo en ADS. (b) Esquema físico

Para ilustrar la importancia del diseño del conector en 'T' se ha simulado el mismo circuito sin tener en cuenta la relación $W_1 > W_3 > W_2$ en la Figura 4.14 y teniéndolo en cuenta en la Figura 4.15. Como puede apreciarse, para aquella simulación donde sí que se ha tenido en cuenta la relación que establece ADS, la diferencia entre las bandas es menor.

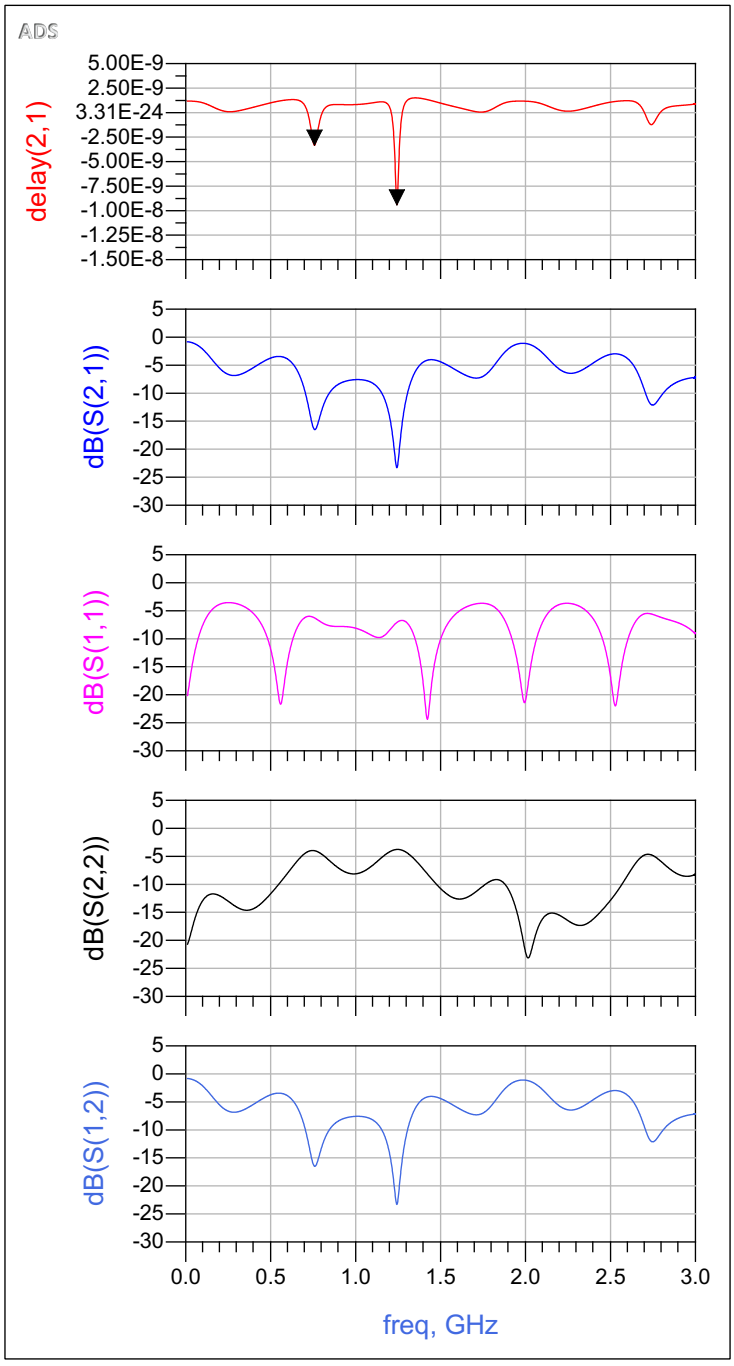


Figura 4.14: Respuesta con el conector 'T' mal configurado (Modelo 1)

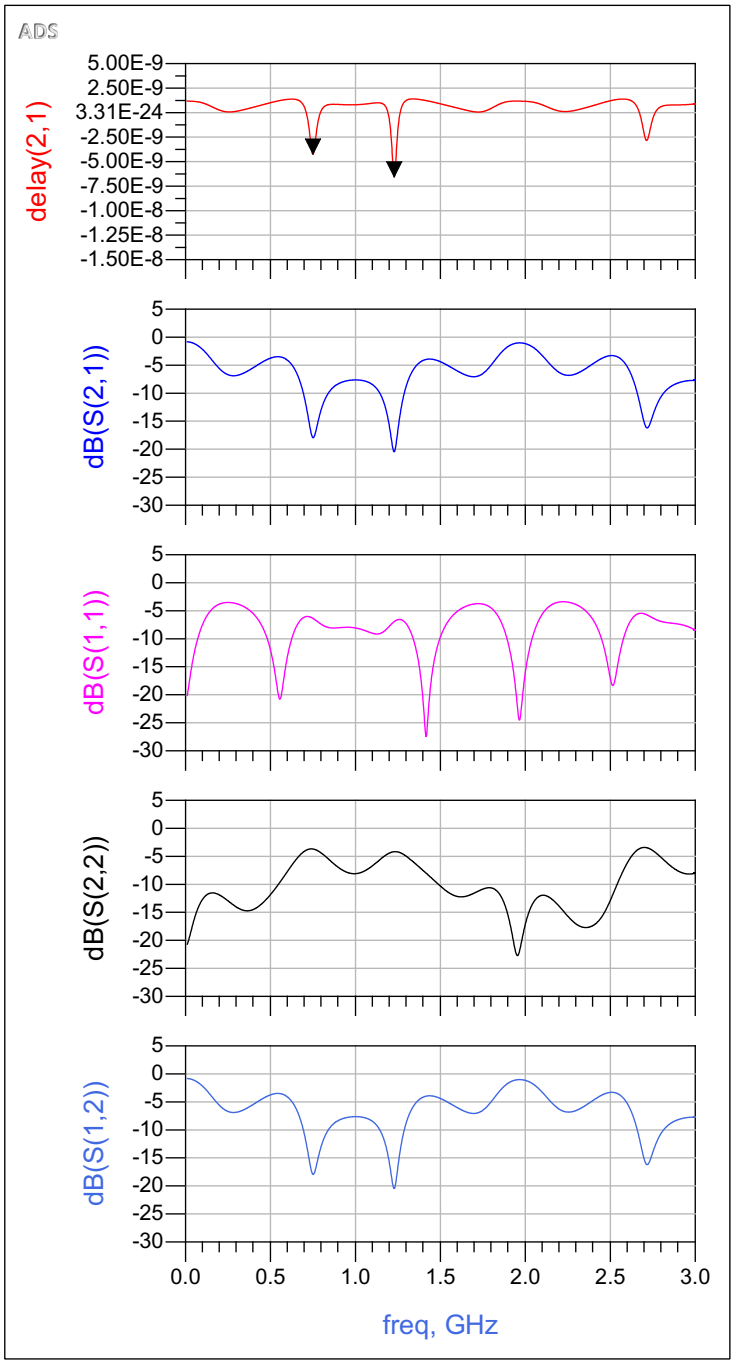


Figura 4.15: Respuesta con el conector 'T' correctamente configurado (Modelo 1)

4.2.3. Simulación de onda completa (Sonnet)

La simulación más exacta es aquella que se hace calculando los campos electromagnéticos en cada punto del circuito y, a su vez, es la más costosa computacionalmente. Sin embargo, llegados al punto previo a la fabricación, es aconsejable realizar este tipo de simulación para conocer con más fiabilidad la respuesta que tendrá el circuito en la vida real.

Este tipo de simulaciones se les conoce con el nombre de onda completa y para llevarlas a cabo en este TFM se utilizará el software Sonnet.

En el Anexo C: Tutorial Sonnet, se explica de forma detallada cómo configurar y crear tiras (*strip*) con este software. Para realizar el estudio de una topología que puede variar las dimensiones físicas de sus líneas también es interesante parametrizar el circuito (Apartado C.2.2.2). Mediante esta técnica es posible variar las propiedades de las líneas de forma rápida y agilizar la simulación de prototipos así como un ajuste fino de los mismos. En este sentido, siguiendo las dimensiones de la tabla 4.5, el prototipo quedaría tal y como se muestra en la Figura 4.16

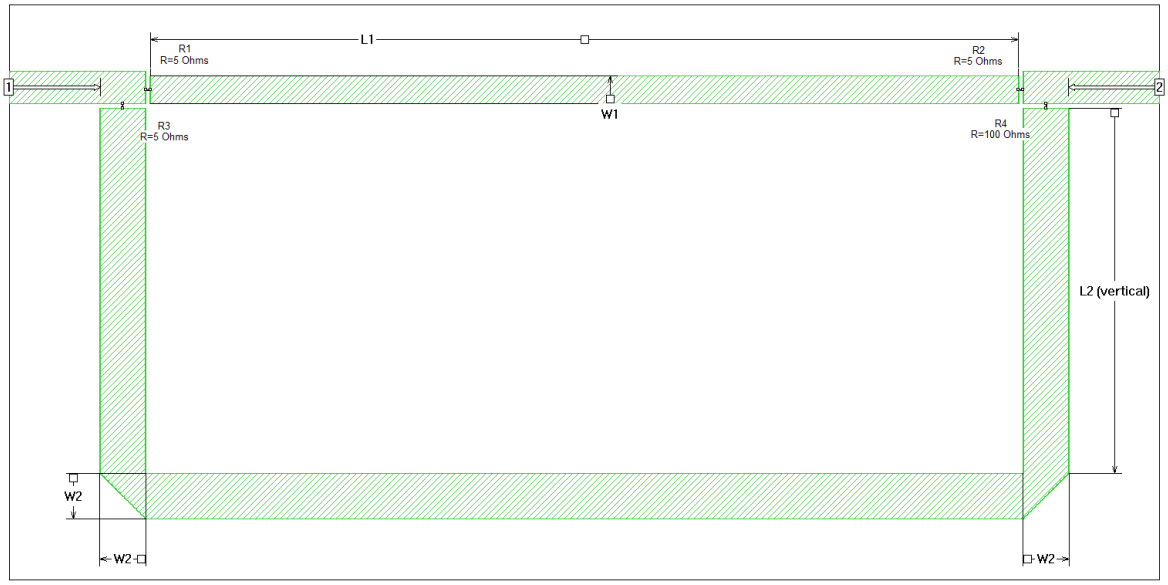


Figura 4.16: Prototipo en Sonnet (Modelo 1)

Tras la correspondiente simulación y configuración de la vista de representación con los parámetros S deseados y el retardo de grupo, se obtiene la respuesta de la topología propuesta en la Figura 4.17.

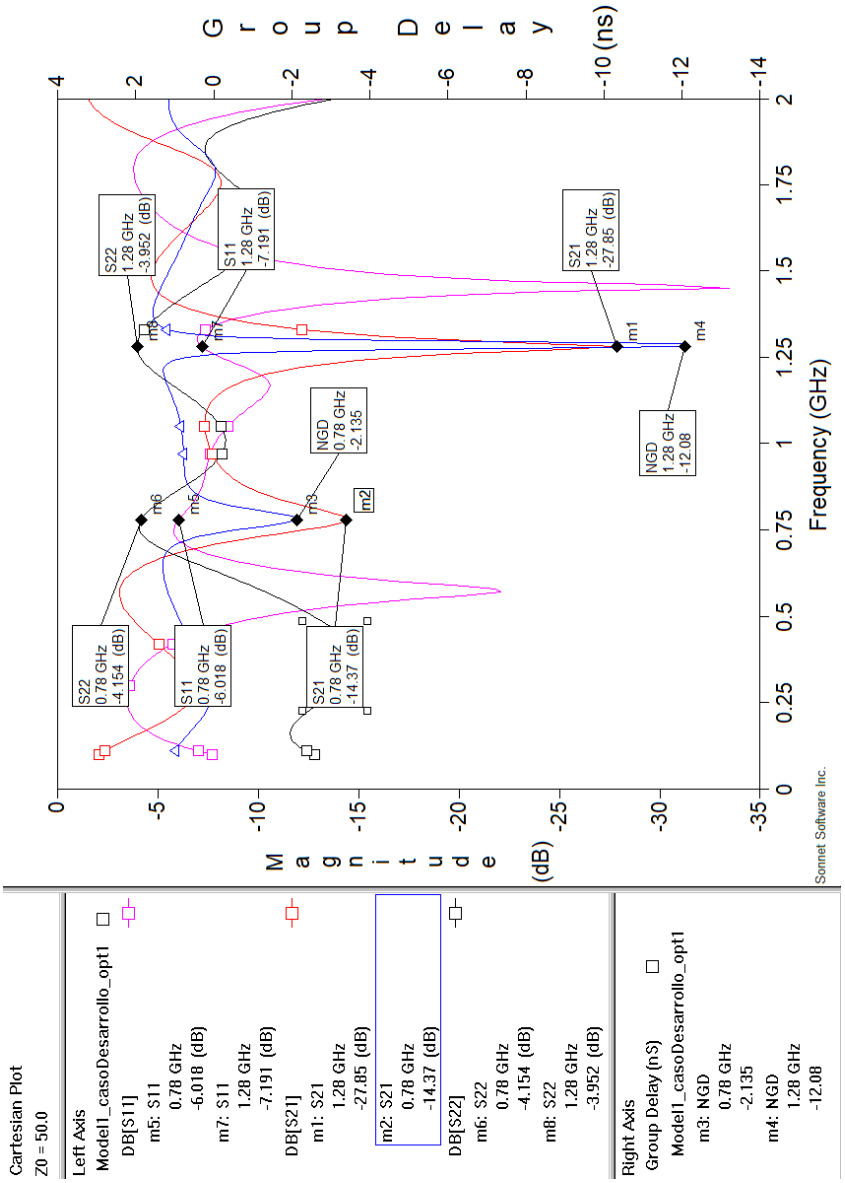


Figura 4.17: Respuesta de la simulación en Sonnet

5. Resultados

En este capítulo se exponen los resultados del estudio de la topología propuesta. En primer lugar se mostrará el análisis parámetrico por fuerza bruta con curvas de diseño de interés. Posteriormente se presentan los resultados principales de los casos escogidos mediante las simulaciones circuitales y de onda completa.

5.1. Análisis paramétrico

Tal y como se ha explicado en el Capítulo 4, la imposibilidad de obtener expresiones analíticamente de la respuesta del circuito nos ha llevado a realizar un análisis paramétrico por fuerza bruta. La computación de los 85.766.121 casos, muy diferentes entre sí, hace que sea complicado encontrar relaciones entre todos ellos.

Teniendo en cuenta que se trata de una topología que ofrece respuestas de una (central), dos (laterales) y tres bandas, se filtran los resultados para evaluar el circuito bajo esas tres condiciones. Además, para que una respuesta se considere apta, la banda o bandas deben tener un $ngd <= -1 \cdot 10^{-9}$ (s) y una atenuación $S_{2,1} >= -20$ (dB).

5.1.1. Modelo 1 - Doble Banda

La condiciones para que sea de Doble Banda se establecen en base a las ya mencionadas, teniendo en cuenta que además para este caso el retardo de grupo de la banda central debe ser positivo o no localizarse un mínimo en dicho punto.

Para analizar este caso, se presenta la Tabla 5.1 con los valores mínimos y máximos para todos los parámetros en relación a las bandas laterales; por otro lado en la Figura 5.1 y 5.2 se representan las distribuciones más interesantes en forma de gráfica; y por último los histogramas de las variables (Figura 5.3).

A partir de este análisis es posible localizar combinaciones para las cuales obtengamos resultados interesantes en función del objetivo. Por ejemplo, se detectaron casos interesantes con $Z_1 = 25$ (Ω) y $Z_1 = 35$ (Ω), por lo que se generaron las siguientes curvas de diseño para el modelo 1 con dichos valores de impedancias y $R_1 = R_3 = 0$ (Ω). En las Figuras 5.19-5.9 se exponen dichas curvas de diseño en función de distintos parámetros.

Ante estas curvas de diseño, podemos seleccionar una combinación concreta y simular rápidamente su respuesta ideal en MATLAB ® mediante el Código D.1, obteniendo la respuesta de la Figura 5.10. Los valores seleccionados para las variables se muestran en la Tabla 5.2.

Parámetro	Mínimo	Máximo
$\boldsymbol{NGD}$	$-19,681$ ns	-1 ns
$\boldsymbol{f_2/f_1}$	$1,4969$	$5,0423$
$\boldsymbol{S_{21}}$	-20 dB	$-8,3247$ dB
$\boldsymbol{S_{11}}$	$-21,8196$ dB	$-0,0505$ dB
$\boldsymbol{S_{22}}$	$-21,8196$ dB	$-0,0505$ dB
$\boldsymbol{BW}$	27 MHz	243 MHz
$\boldsymbol{BW_{1dB}}$	8 MHz	233 MHz
$\boldsymbol{BW_{3dB}}$	14 MHz	417 MHz

Tabla 5.1: Análisis numérico del Modelo 1 - Doble Banda

$R_1(\Omega)$	$Z_1(\Omega)$	$R_2(\Omega)$	$R_3(\Omega)$	$Z_2(\Omega)$	$R_4(\Omega)$
0	25	10	0	35	80

Tabla 5.2: Valores seleccionados para las pruebas del Modelo 1 - Doble Banda

Por lo que obtenemos una configuración con las características de la Tabla 5.3.

Parámetro	Valor
$\boldsymbol{NGD}$	$-2,5056$ ns
$\boldsymbol{f_2/f_1}$	$1,6178$
$\boldsymbol{S_{21}}$	$-14,276$ dB
$\boldsymbol{S_{11}}$	$-4,0813$ dB
$\boldsymbol{S_{22}}$	$-3,6170$ dB
$\boldsymbol{BW}$	103 MHz
$\boldsymbol{BW_{1dB}}$	55 MHz
$\boldsymbol{BW_{3dB}}$	112 MHz

Tabla 5.3: Valor de los parámetros para el caso elegido del Modelo 1 - Doble Banda

Para poder simular en onda completa esta configuración es necesario calcular las dimensiones físicas de las líneas de transmisión para los substratos Rogers 4003 y TLX-8, las cuales se muestran en la Tabla 5.4.

	Eléctricos		**Rogers 4003**		**TLX-8**	
Línea de Transmisión	$Z_0(\Omega)$	$\theta_0(\Omega)$	W(mm)	L(mm)	W(mm)	L(mm)
Superior (TL1)	25	180°	$9,03$	$86,09$	$5,65$	$99,74$
Inferior (TL2)	35	360°	$5,75$	$175,66$	$3,64$	$202,61$
Puertos	50		$3,36$		$2,17$	

Tabla 5.4: Dimensiones físicas para el caso bajo estudio del Modelo 1 - Doble Banda

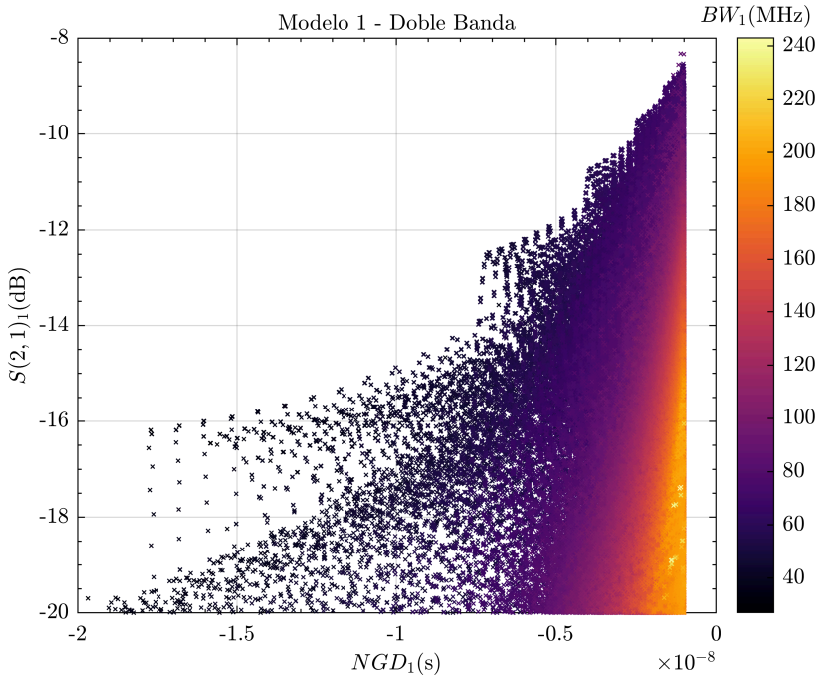


Figura 5.1: NGD frente a atenuación y ancho de banda (Análisis Modelo 1 - Doble Banda)

Por lo tanto, el diseño implementado en Sonnet sólo tiene dos resistencias (junto al puerto de salida) y es necesario ajustar la conexión en 'T' del puerto de entrada para conseguir unos resultados óptimos. En la Figura 5.11 se muestra un posible diseño de la conexión en 'T' para el substrato Rogers 4003. También se han evaluado otros, centrando el puerto 1 frente a la línea de transmisión 1, pero con peores resultados. Esto se debe a los efectos parásitos que suponen este tipo de uniones. La respuesta de este prototipo se muestra en la Figura 5.12. En las Figuras 5.13 y 5.14 se muestra el mismo prototipo para el substrato TLX-8.

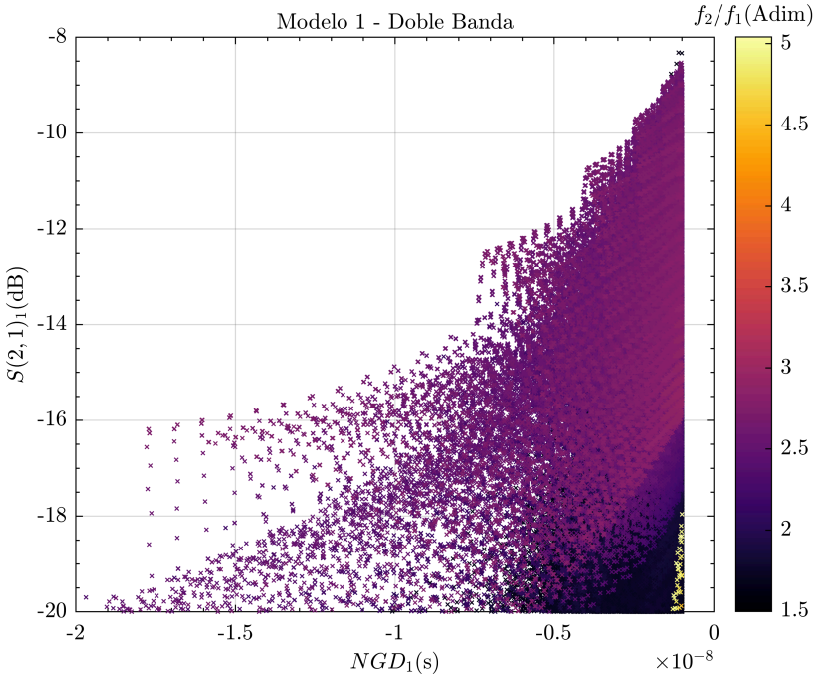


Figura 5.2: NGD frente a atenuación y relación entre bandas (Análisis Modelo 1 - Doble Banda)

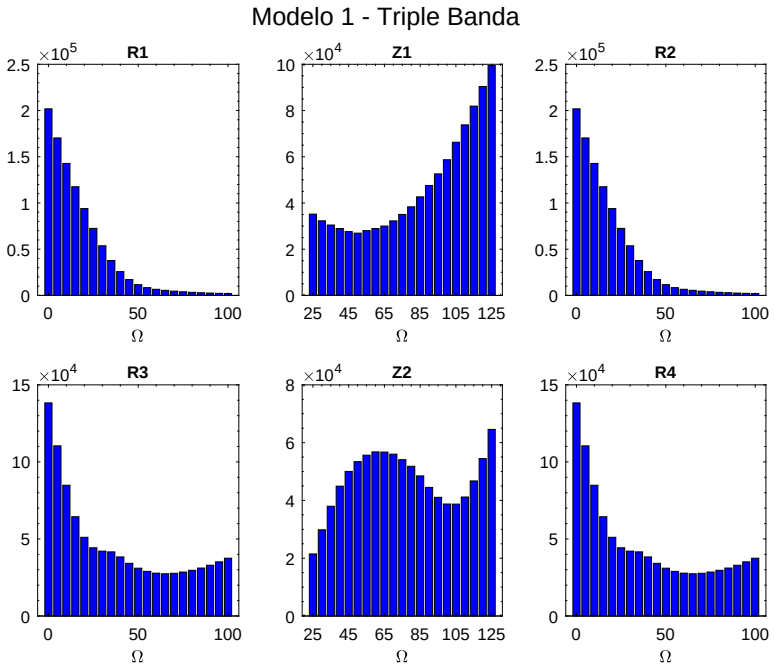


Figura 5.3: Histogramas de las variables para el Modelo 1 - Doble Banda

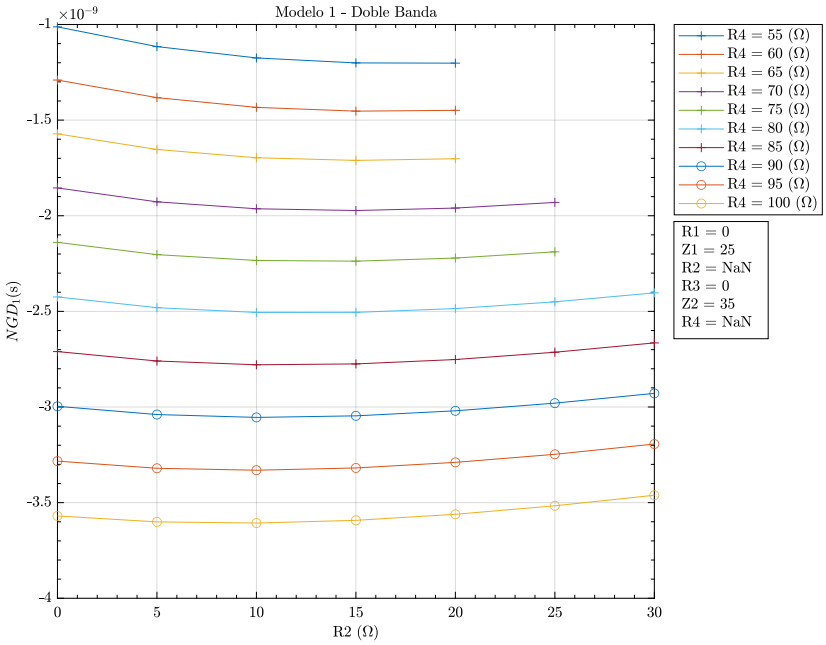


Figura 5.4: Curva de diseño en función de NGD

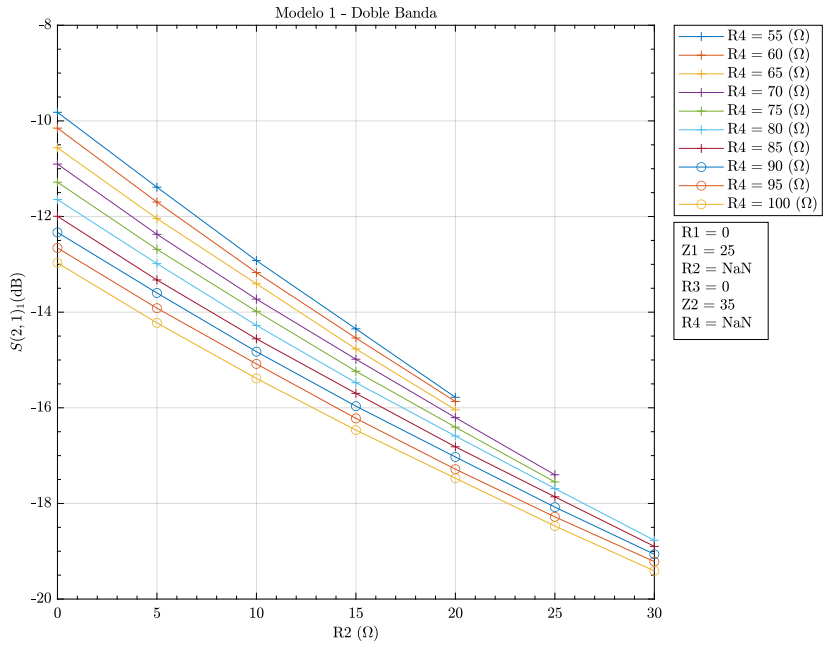


Figura 5.5: Curva de diseño en función del parámetro $S(2,1)$

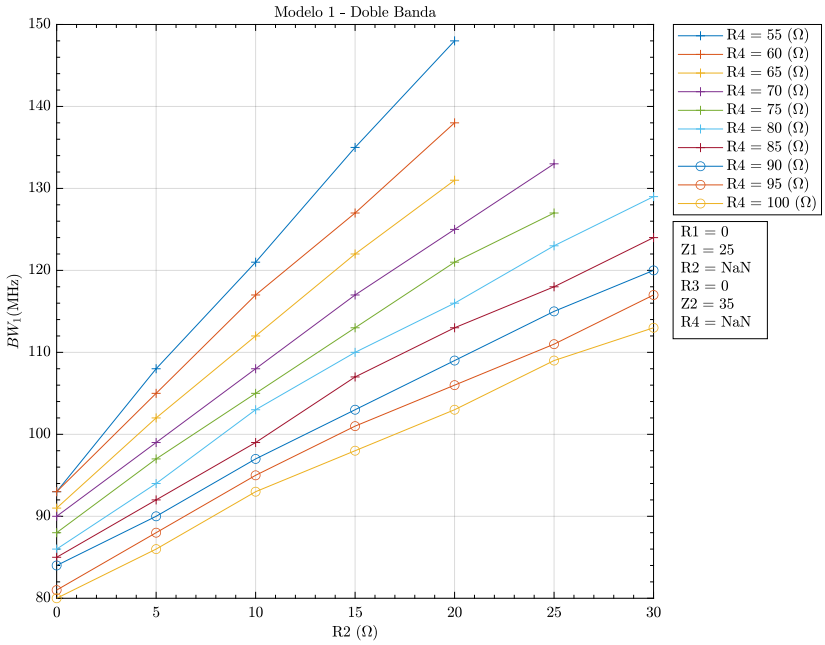


Figura 5.6: Curva de diseño en función del ancho de banda

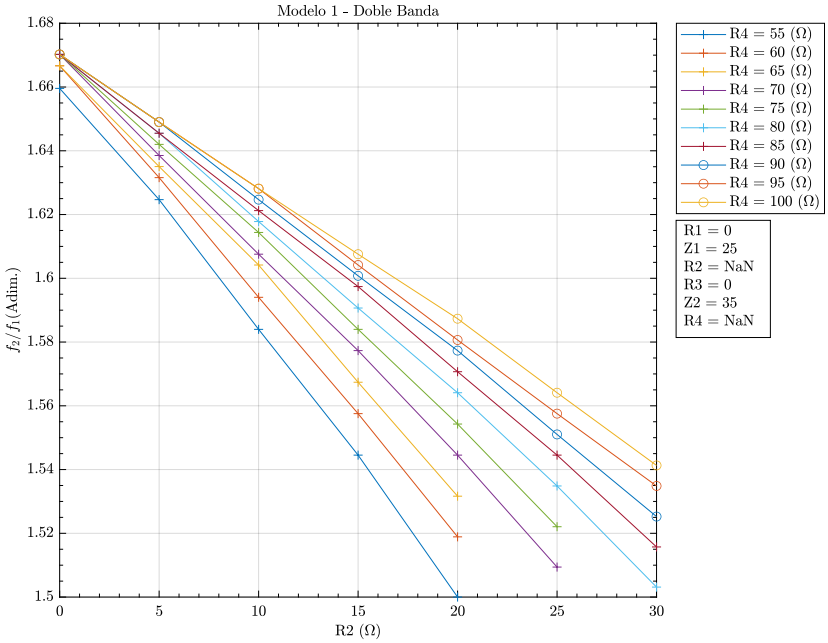


Figura 5.7: Curva de diseño en función de la relación entre bandas laterales

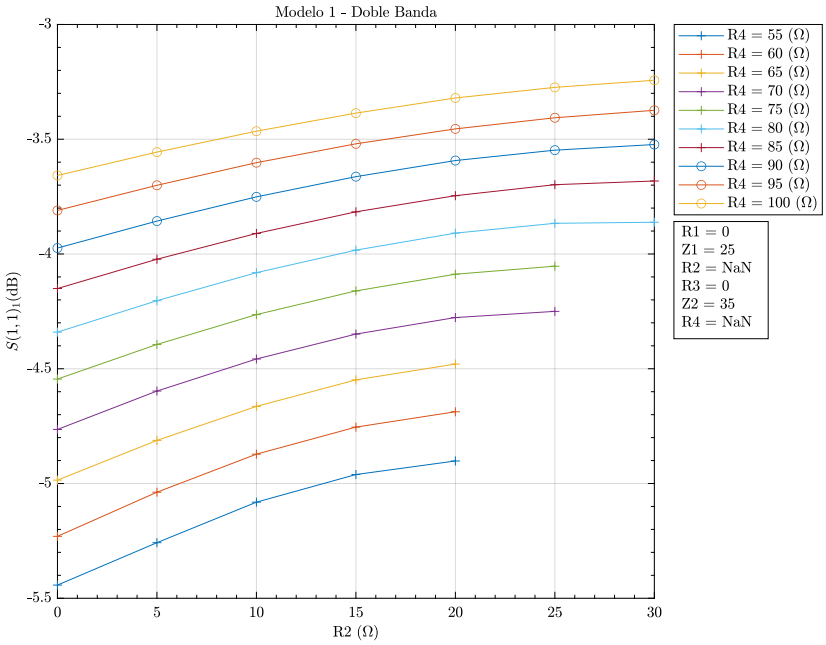


Figura 5.8: Curva de diseño en función del parámetro $S(1,1)$

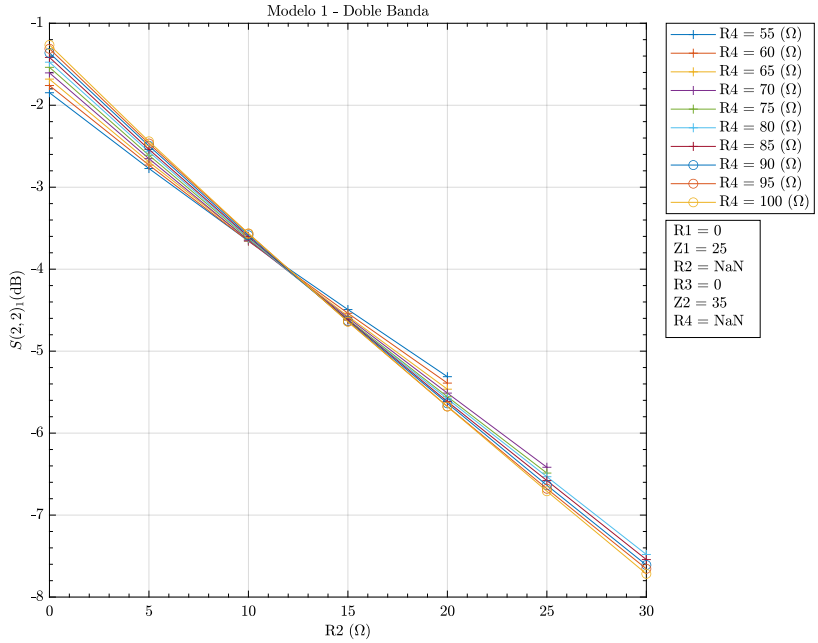


Figura 5.9: Curva de diseño en función del parámetro $S(2,2)$

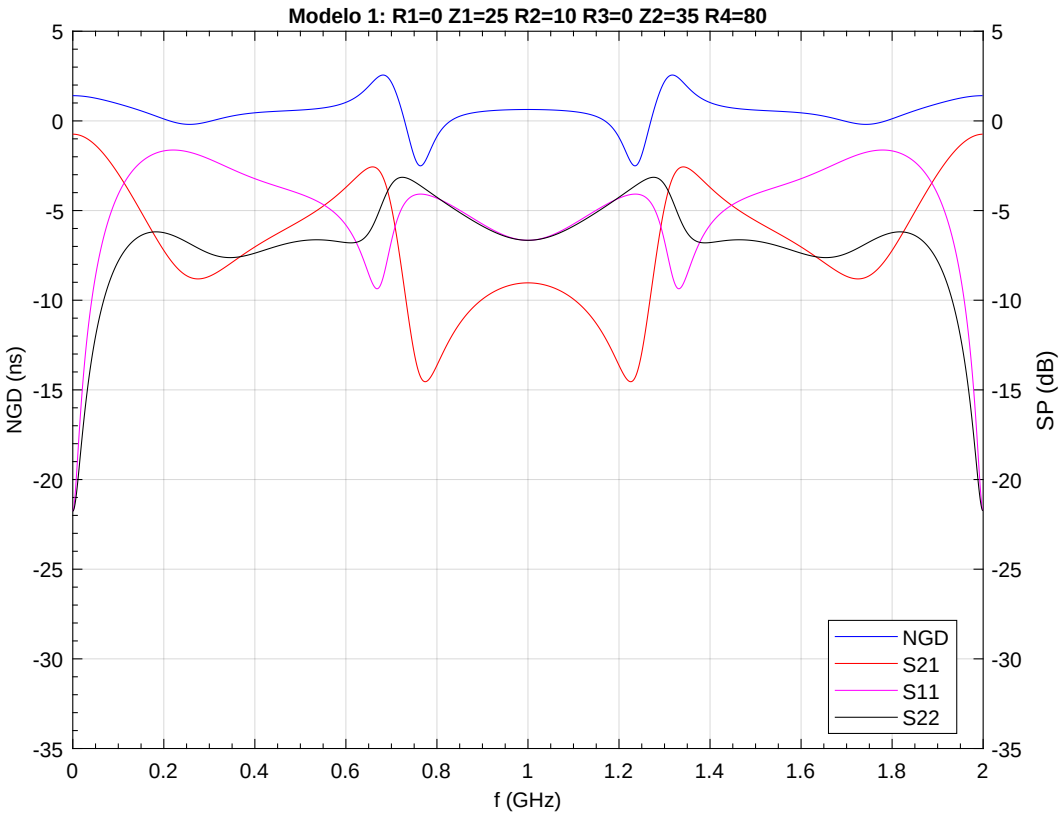


Figura 5.10: Respuesta ideal para la configuración seleccionada

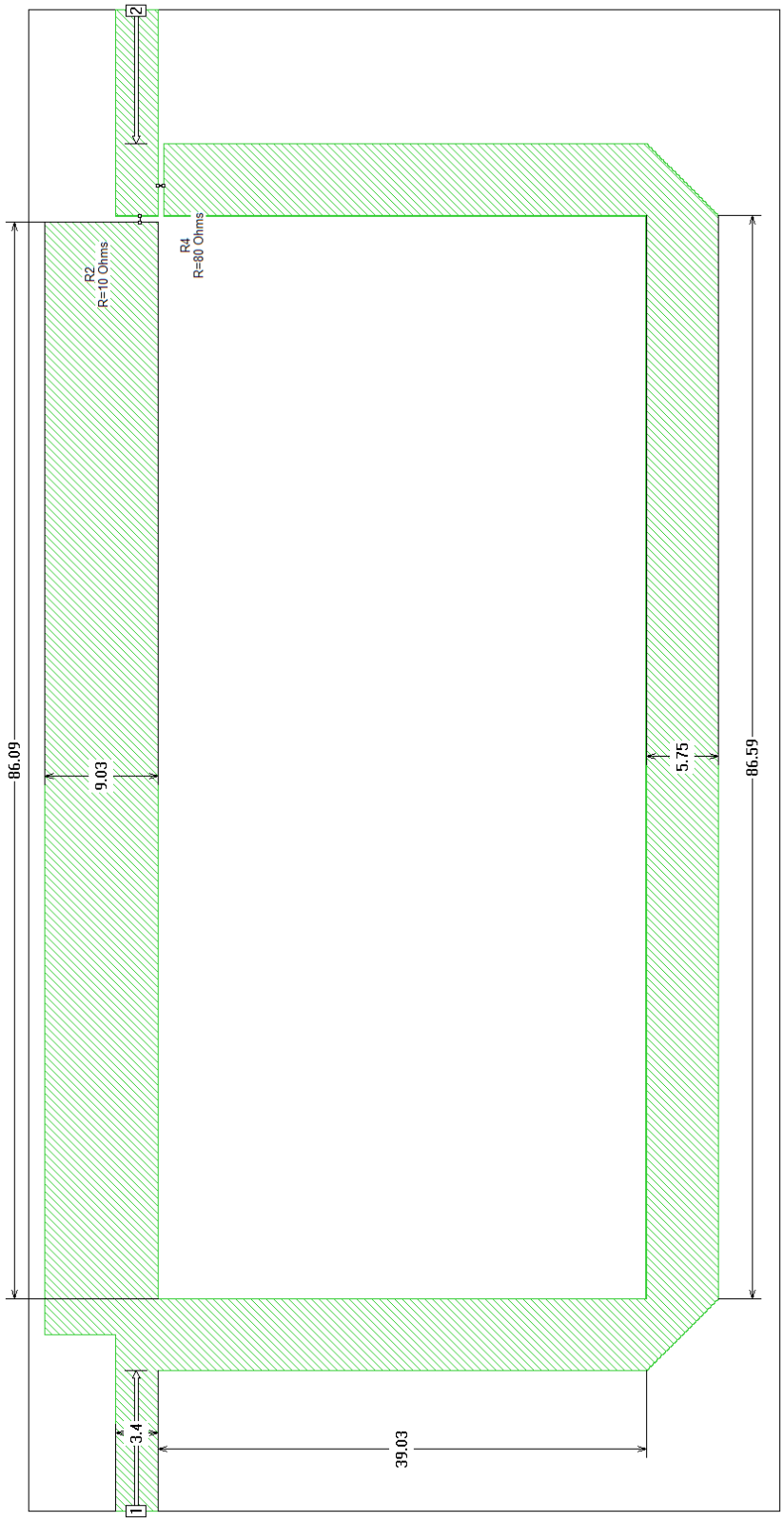


Figura 5.11: Diseño del prototipo en Sonnet (Rogers 4003)

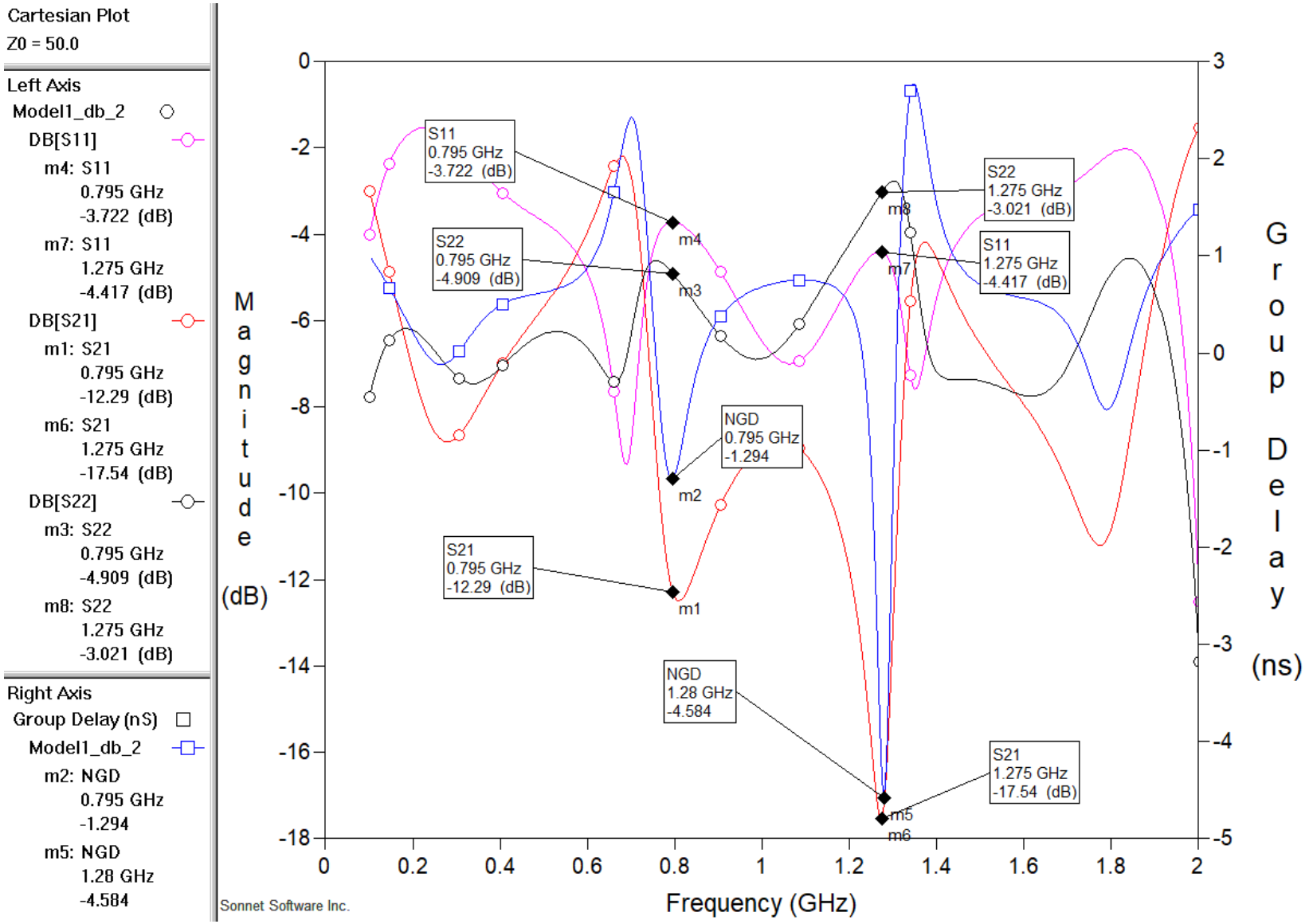


Figura 5.12: Respuesta del prototipo en Sonnet (Rogers 4003)

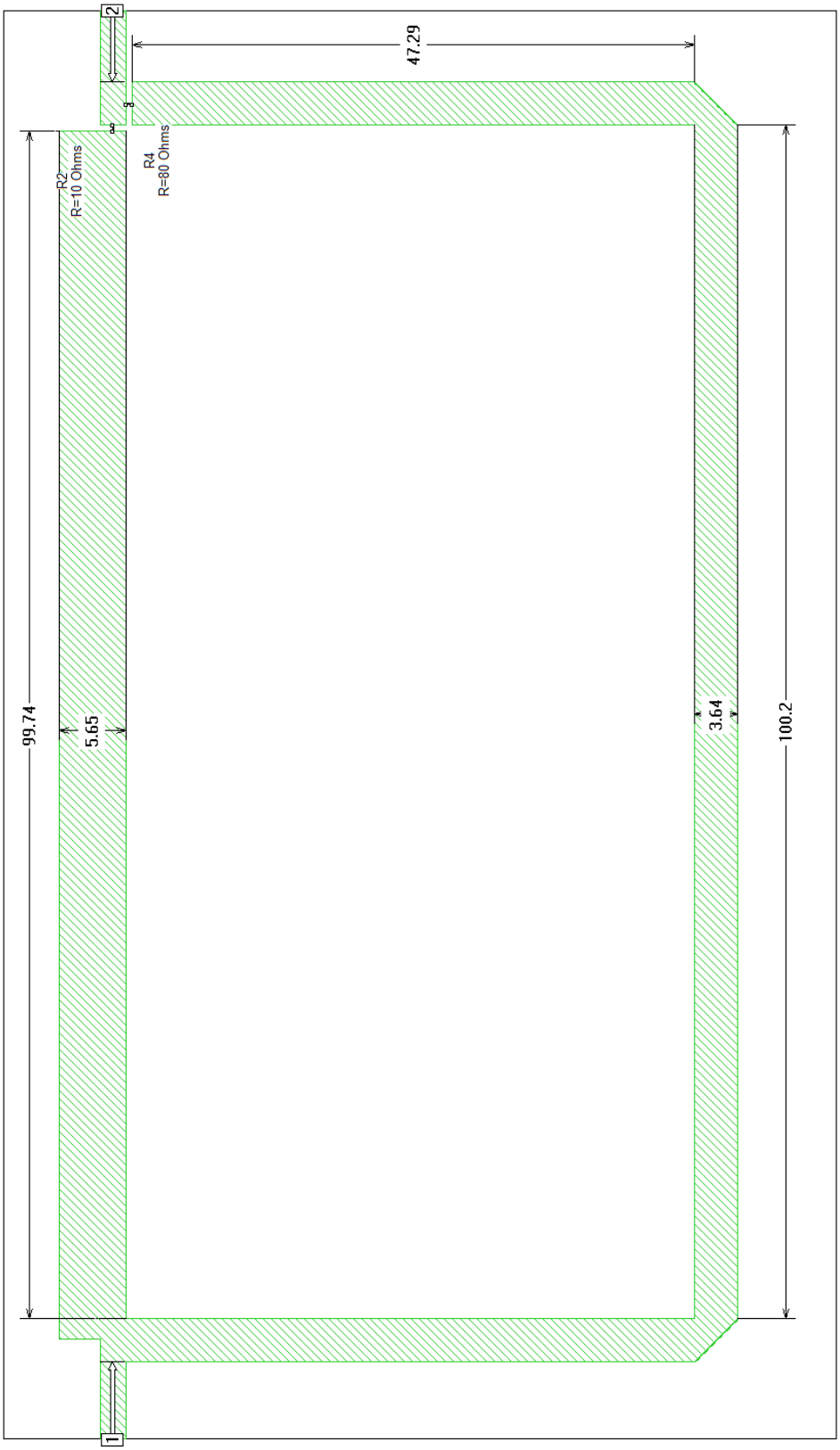


Figura 5.13: Diseño del prototipo en Sonnet (TLX-8)

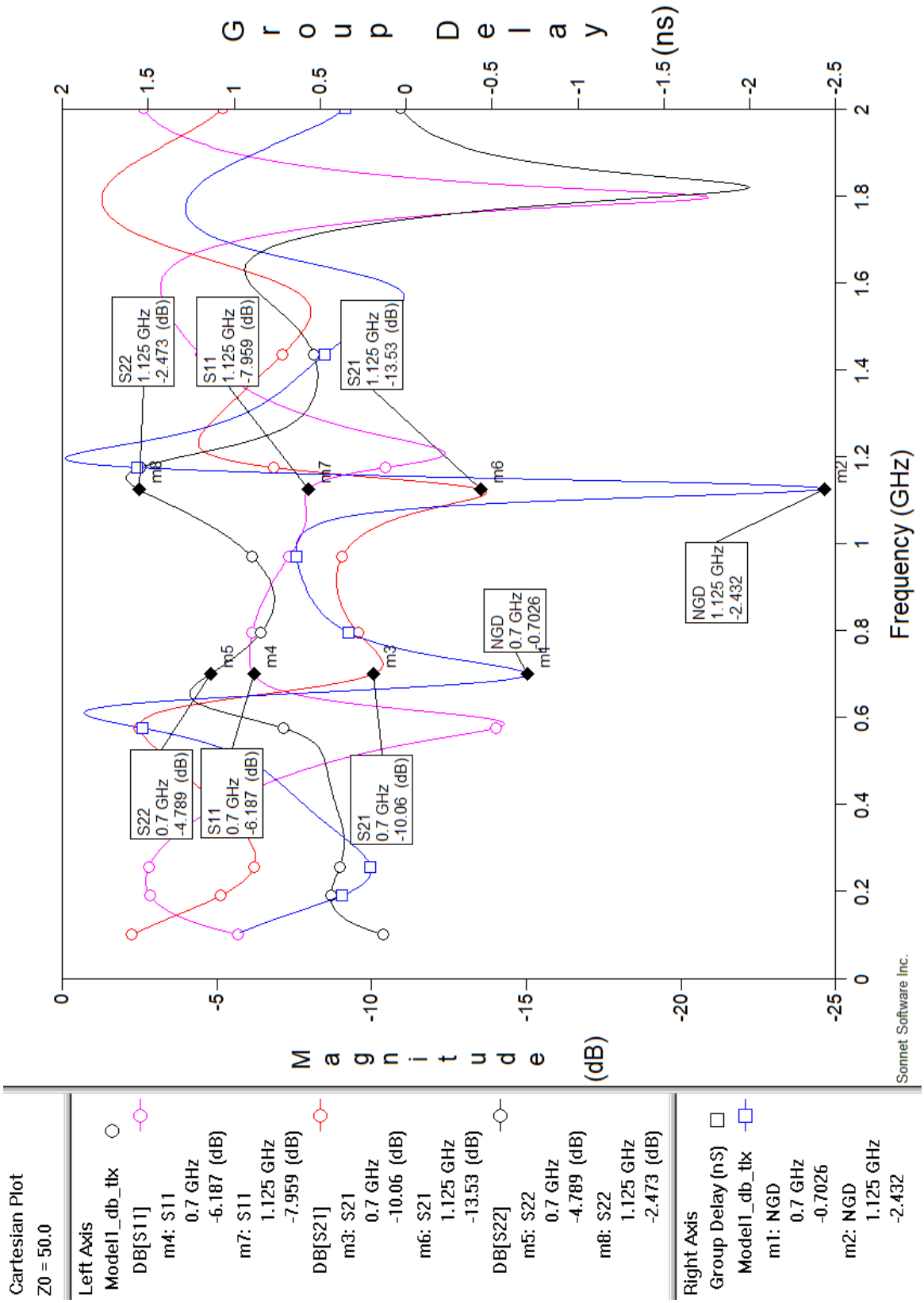


Figura 5.14: Respuesta del prototipo en Sonnet (TLX-8)

Por lo tanto, los valores obtenidos para ambas bandas tras la simulación de onda completa difieren de los calculados idealmente y se muestran en la Tabla 5.5.

	Banda 1			Banda 2		
Parámetro	**Ideal**	**Rogers 4003**	**TLX-8**	**Ideal**	**Rogers 4003**	**TLX-8**
$\boldsymbol{NGD}$ **(ns)**	$-2,506$	$-1,294$	$-0,70$	$-2,506$	$-4,584$	$-2,43$
$\boldsymbol{f}$ **(MHz)**	764	795	700	1.236	1.275	1.125
$\boldsymbol{S_{21}}$ **(dB)**	$-14,28$	$-12,29$	$-10,06$	$-14,28$	$-17,54$	$-13,53$
$\boldsymbol{S_{11}}$ **(dB)**	$-4,08$	$-3,72$	$-6,19$	$-4,08$	$-4,42$	$-7,96$
$\boldsymbol{S_{22}}$ **(dB)**	$-3,62$	$-4,09$	$-4,79$	$-3,62$	$-3,02$	$-2,473$
$\boldsymbol{BW}$ **(MHz)**	103	100	100	103	85	80

Tabla 5.5: Comparación entre la respuesta ideal y la simulada en onda completa

5.1.2. Modelo 1 - Triple Banda

Las condiciones para que se de la configuración Triple Banda se consisten en aplicar tanto a las bandas laterales como la central las condiciones de $ngd <= -1 \cdot 10^{-9}$ (s) y $S_{2,1} >= -20$ (dB). De esta forma es posible analizar paramétricamente el sistema. Al igual que con el caso de Doble Banda, se presentan en la Tabla 5.6 los valores mínimos y máximos para todos los parámetros de interés. También se adjuntan las distribuciones más interesantes tanto para las bandas laterales (indicado con el subíndice 1) como para la banda central (subíndice C) en las Figuras 5.15 y 5.16 respectivamente. En la Figura 5.17 se muestran los histogramas de las variables para las configuraciones Triple Banda

Por lo tanto, es posible encontrar curvas de diseño interesantes. En este sentido para $R_1 = R_2 = 5$ (Ω), $R_3 = 100$ (Ω) y $R_4 = 15$ (Ω) se encuentran casos de interés variando los valores de las impedancias superior e inferior. En las Figuras 5.18 - 5.29 se muestran las curvas de diseño pertinentes. Tal y como se observa, las curvas que describen las bandas centrales son más irregulares, por lo que aunque es relativamente sencillo encontrar un triple banda con NGD similares para sus tres bandas, es algo más complicado encontrar atenuaciones ($S(2,1)$) parecidas.

Tras seleccionar una combinación concreta, se obtiene la respuesta ideal en MATLAB ® de la Figura 5.30. Los valores seleccionados para esta combinación se muestran en la Tabla 5.7. Y se obtiene una configuración con las características de la Tabla 5.8.

Parámetro	Mínimo	Máximo
$\boldsymbol{NGD_{1,2}}$	$-8,3681$ ns	-1 ns
$\boldsymbol{NGD_C}$	$-6,2775$ ns	-1 ns
$\boldsymbol{f_2/f_1}$	$1,2962$	$2,8095$
$\boldsymbol{S_{21_{1,2}}}$	$-19,9913$ dB	$-5,8674$ dB
$\boldsymbol{S_{21_C}}$	$-19,9740$ dB	$-10,8818$ dB
$\boldsymbol{S_{11_{1,2}}}$	$-8,1144$ dB	$-0,3491$ dB
$\boldsymbol{S_{11_C}}$	$-6,2014$ dB	$-1,2136$ dB
$\boldsymbol{S_{22_{1,2}}}$	$-8,1144$ dB	$-0,3491$ dB
$\boldsymbol{S_{22_C}}$	$-6,2014$ dB	$-1,2136$ dB
$\boldsymbol{BW_{1,2}}$	53 MHz	109 MHz
$\boldsymbol{BW_C}$	93 MHz	213 MHz
$\boldsymbol{BW_{1dB_{1,2}}}$	17 MHz	203 MHz
$\boldsymbol{BW_{1dB_C}}$	22 MHz	90 MHz
$\boldsymbol{BW_{3dB_{1,2}}}$	33 MHz	384 MHz
$\boldsymbol{BW_{3dB_C}}$	44 MHz	176 MHz

Tabla 5.6: Análisis numérico del Modelo 1 - Triple Banda

$R_1(\Omega)$	$Z_1(\Omega)$	$R_2(\Omega)$	$R_3(\Omega)$	$Z_2(\Omega)$	$R_4(\Omega)$
5	95	5	10	125	15

Tabla 5.7: Valores seleccionados para las pruebas del Modelo 1 - Triple Banda

Parámetro	Valor
$\boldsymbol{NGD_{1,2}}$	$-3,003$ ns
$\boldsymbol{NGD_C}$	$-3,697$ ns
$\boldsymbol{f_2/f_1}$	$2,2$
$\boldsymbol{S_{21_{1,2}}}$	$-13,92$ dB
$\boldsymbol{S_{21_C}}$	$-19,24$ dB
$\boldsymbol{S_{11_{1,2}}}$	$-2,467$ dB
$\boldsymbol{S_{11_C}}$	$-2,986$ dB
$\boldsymbol{S_{22_{1,2}}}$	$-2,205$ dB
$\boldsymbol{S_{22_C}}$	$-2,986$ dB
$\boldsymbol{BW_{1,2}}$	85 MHz
$\boldsymbol{BW_C}$	101 MHz
$\boldsymbol{BW_{1dB_{1,2}}}$	36 MHz
$\boldsymbol{BW_{1dB_C}}$	34 MHz
$\boldsymbol{BW_{3dB_{1,2}}}$	72 MHz
$\boldsymbol{BW_{3dB_C}}$	64 MHz

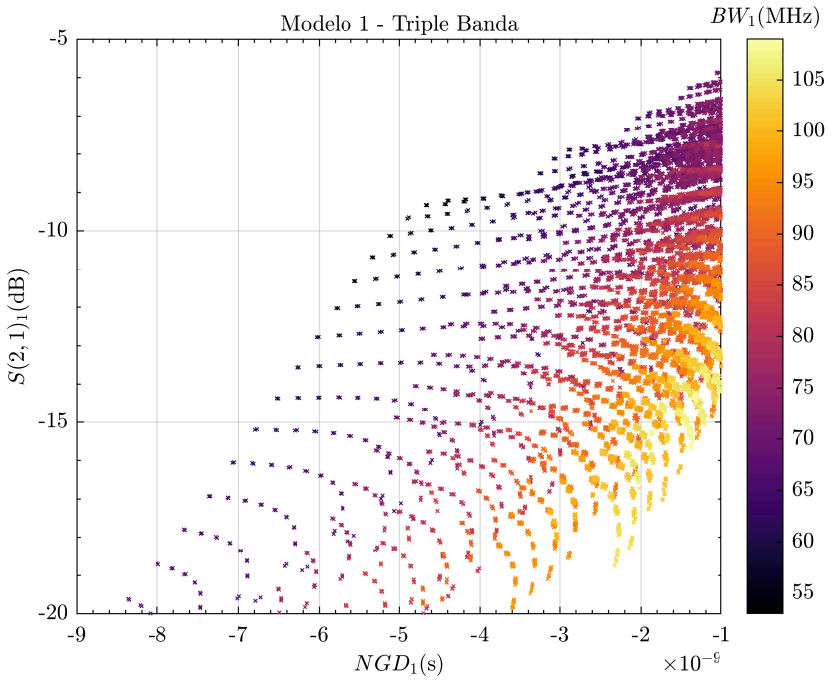
Tabla 5.8: Valor de los parámetros para el caso elegido del Modelo 1 - Doble Banda

Por otro lado, para realizar la simulación en onda completa, se sintetizan las impedancias para obtener sus equivalentes en longitud y anchura, tal y como se muestra en la Tabla 5.9.

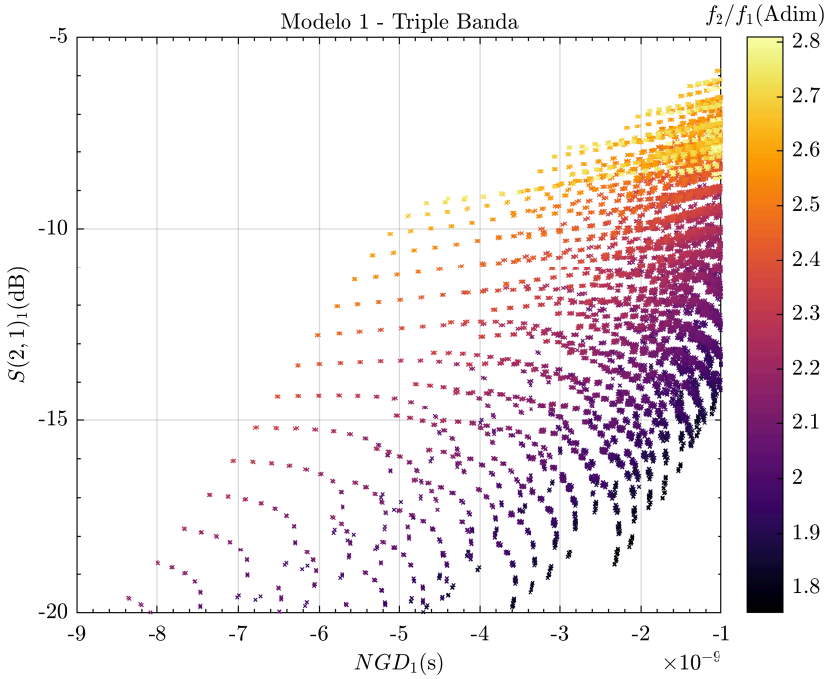
	Eléctricos		**Rogers 4003**		**TLX-8**	
Línea de Transmisión	$Z_0(\Omega)$	$\theta_0(\Omega)$	W(mm)	L(mm)	W(mm)	L(mm)
Superior (TL1)	95	180°	$0,93$	$94,13$	$0,65$	$107,25$
Inferior (TL2)	125	360°	$0,41$	$191,81$	$0,31$	$218,12$
Puertos	50		$3,36$		$2,17$	

Tabla 5.9: Dimensiones físicas para el caso bajo estudio del Modelo 1 - Triple Banda

Al igual que en el apartado anterior, se ha simulado dicho en onda completa mediante el software Sonnet. Los prototipos y las respuestas tanto para el substrato Rogers 4003 como para el TLX-8 se muestran en las Figuras 5.31 - 5.34.

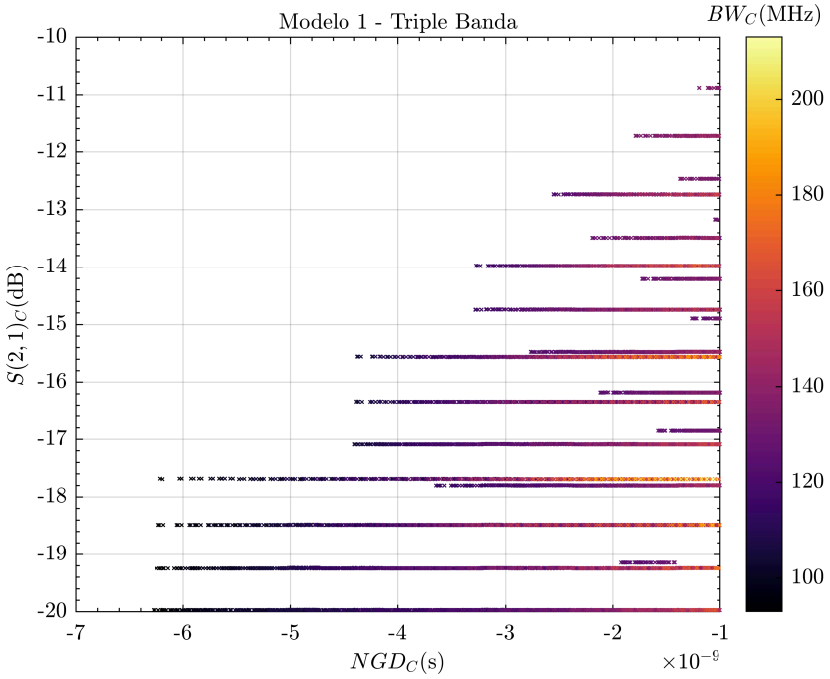


(a) NGD frente a atenuación y ancho de banda.

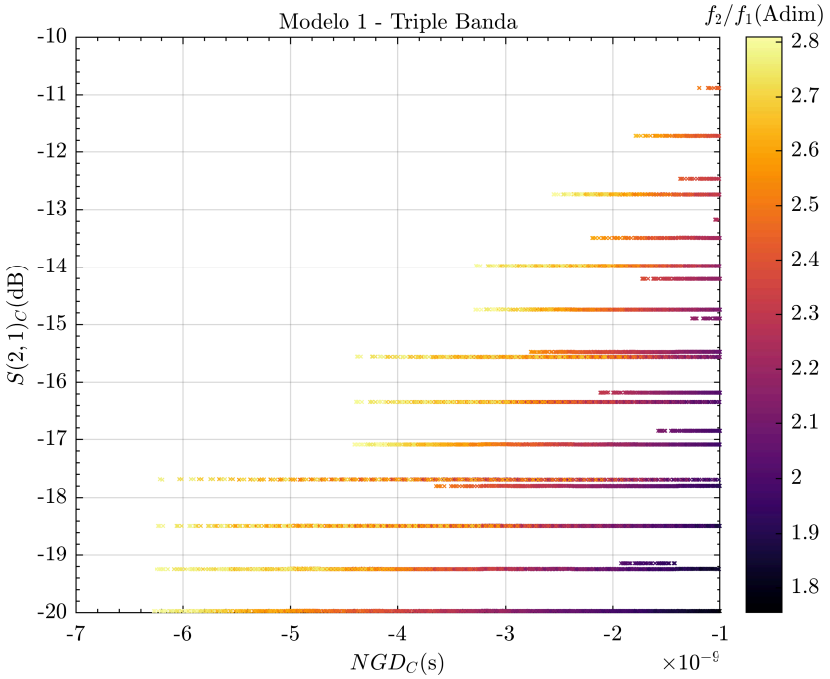


(b) NGD frente a atenuación y relación entre bandas.

Figura 5.15: Análisis Modelo 1 - Triple Banda para las bandas laterales



(a) NGD frente a atenuación y ancho de banda.



(b) NGD frente a atenuación y relación entre bandas.

Figura 5.16: Análisis Modelo 1 - Triple Banda para la banda central

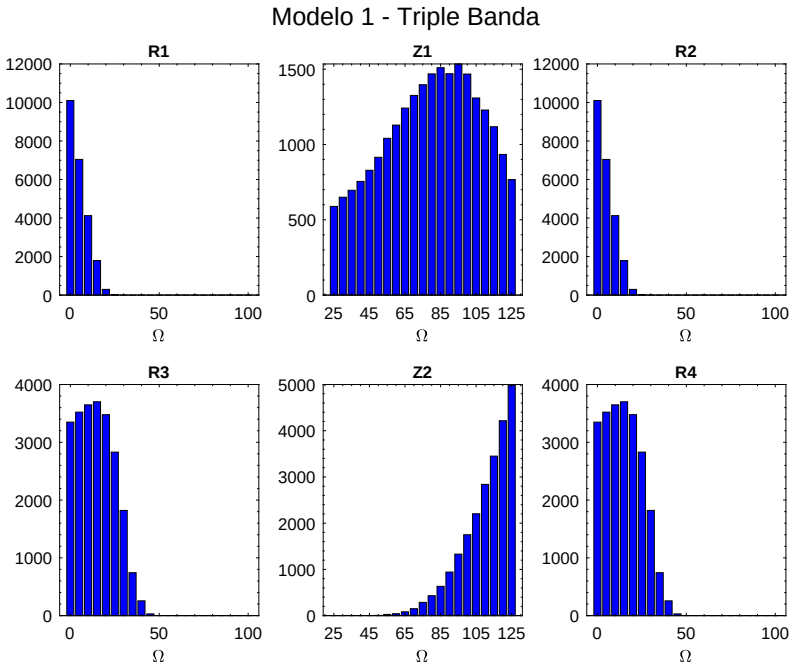


Figura 5.17: Histogramas de las variables para el Modelo 1 - Triple Banda

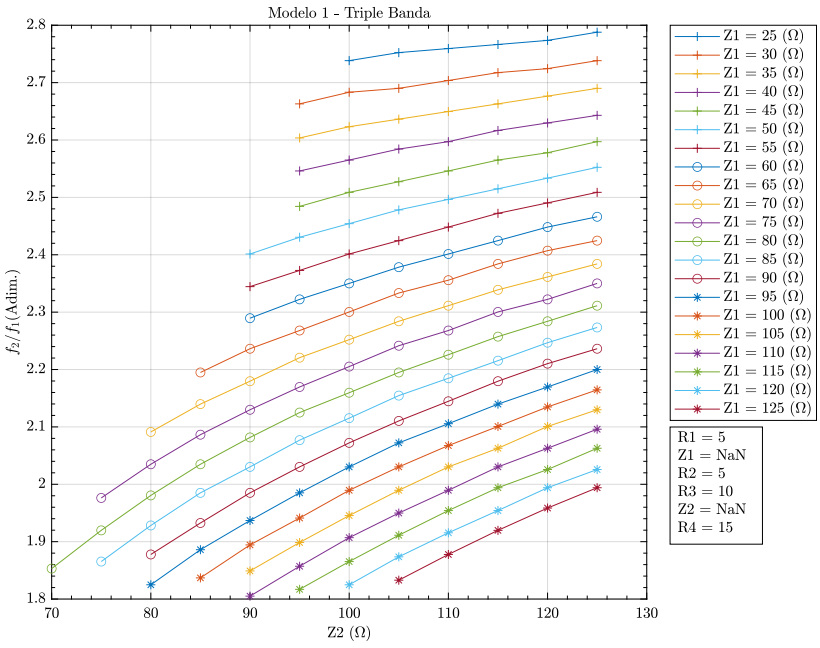


Figura 5.18: Curva de diseño en función de la relación entre bandas laterales

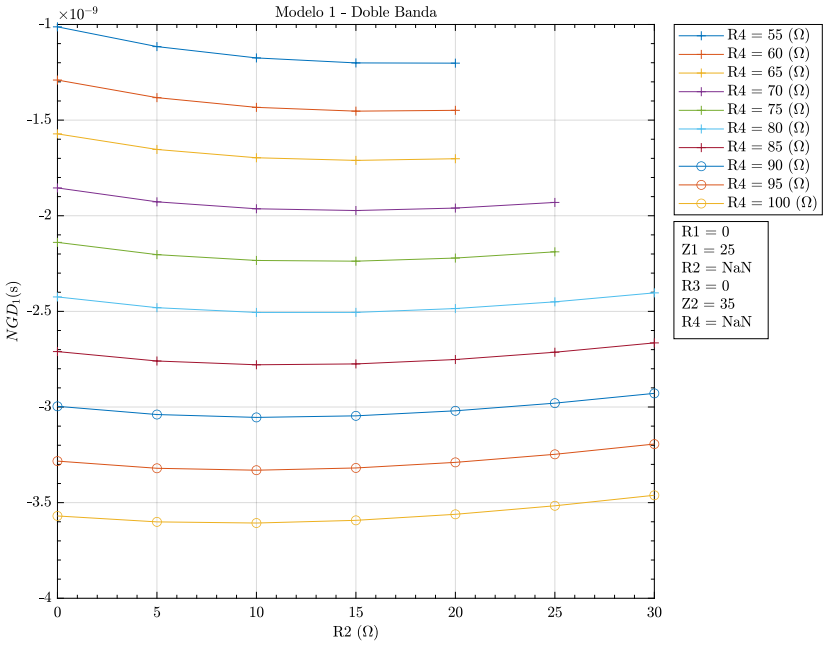


Figura 5.19: Curva de diseño en función de NGD

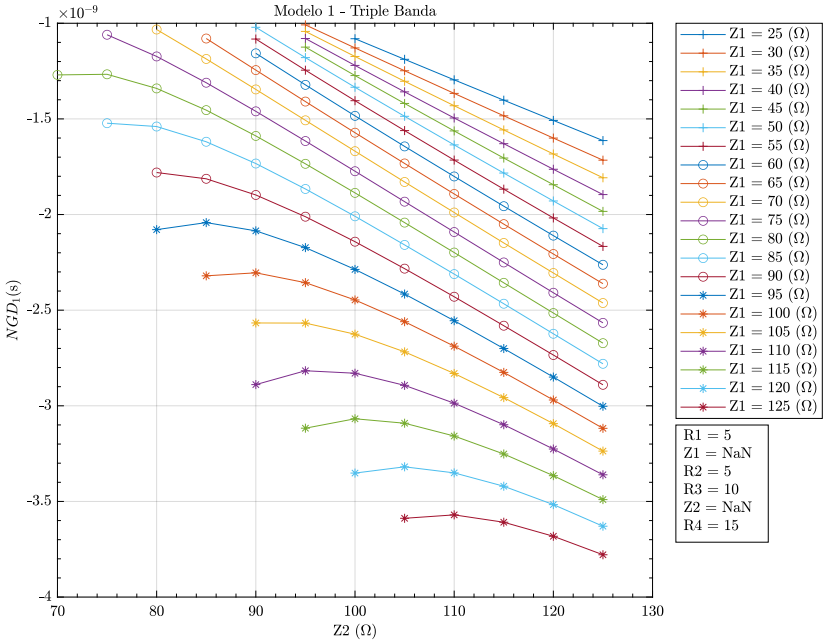


Figura 5.20: Curva de diseño en función de NGD de las bandas laterales

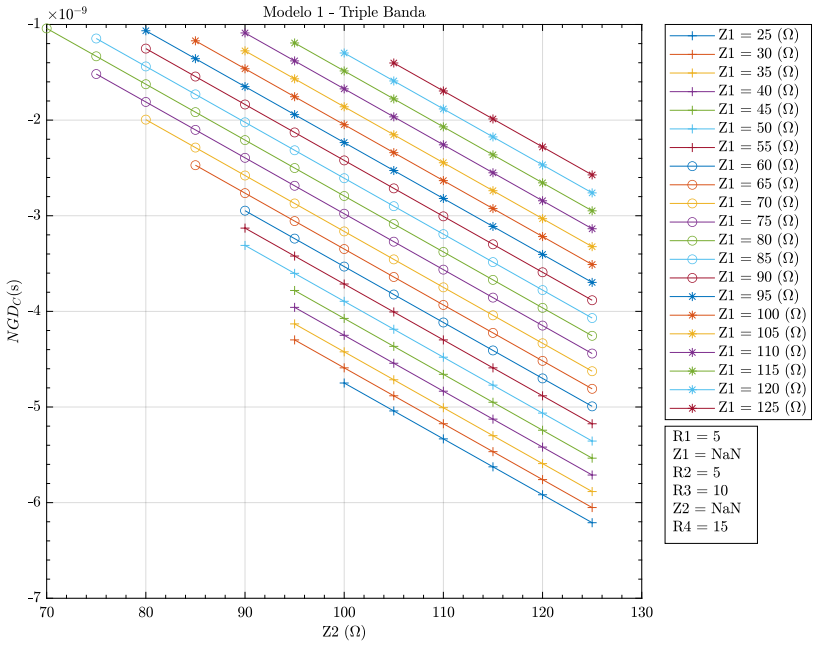


Figura 5.21: Curva de diseño en función de NGD de la banda central

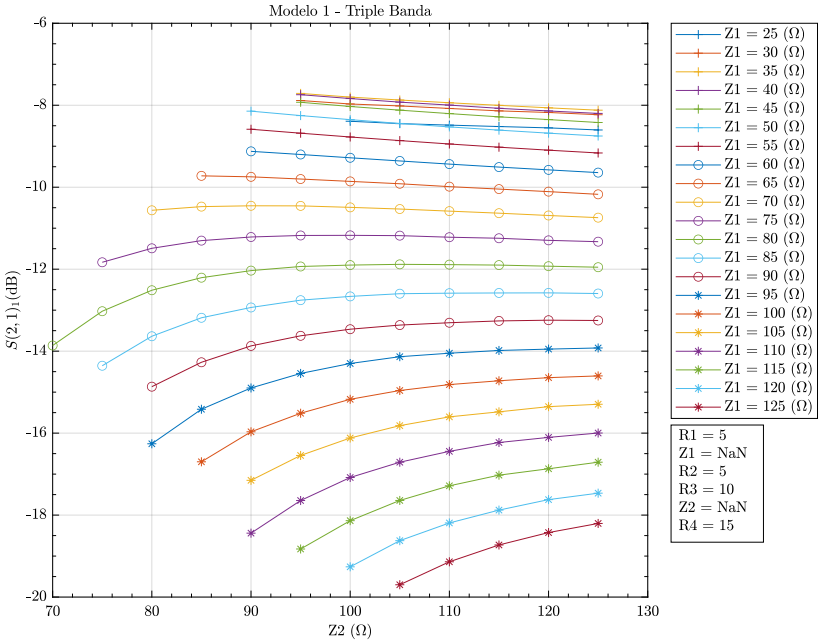


Figura 5.22: Curva de diseño en función del parámetro $S(2,1)$ de las bandas laterales

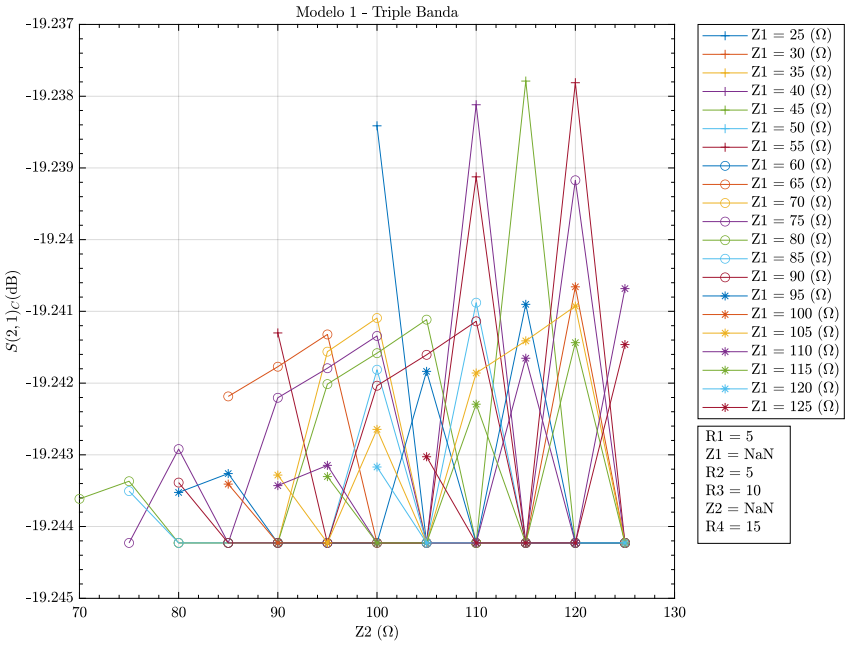


Figura 5.23: Curva de diseño en función del parámetro $S(2,1)$ de la banda central

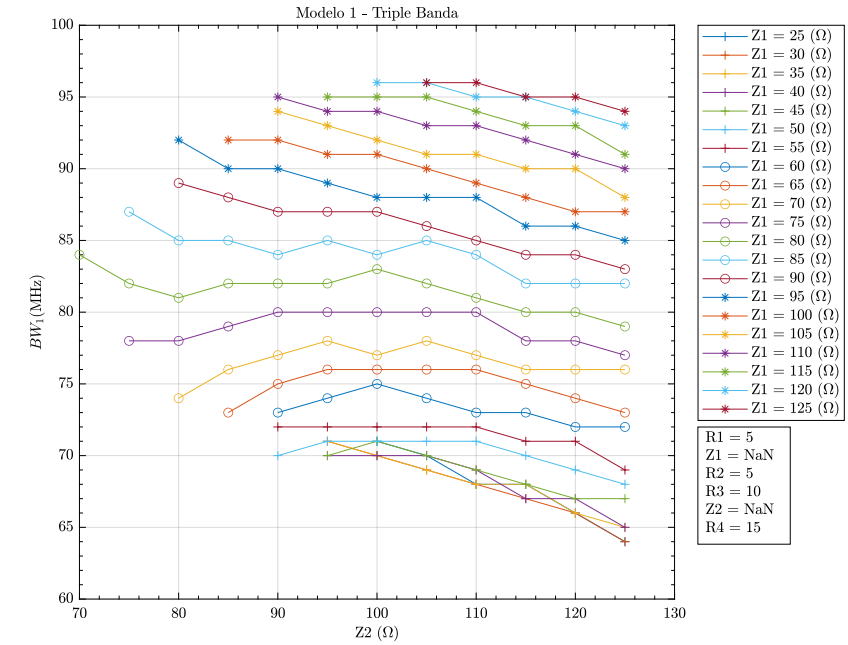


Figura 5.24: Curva de diseño en función del ancho de banda de las bandas laterales

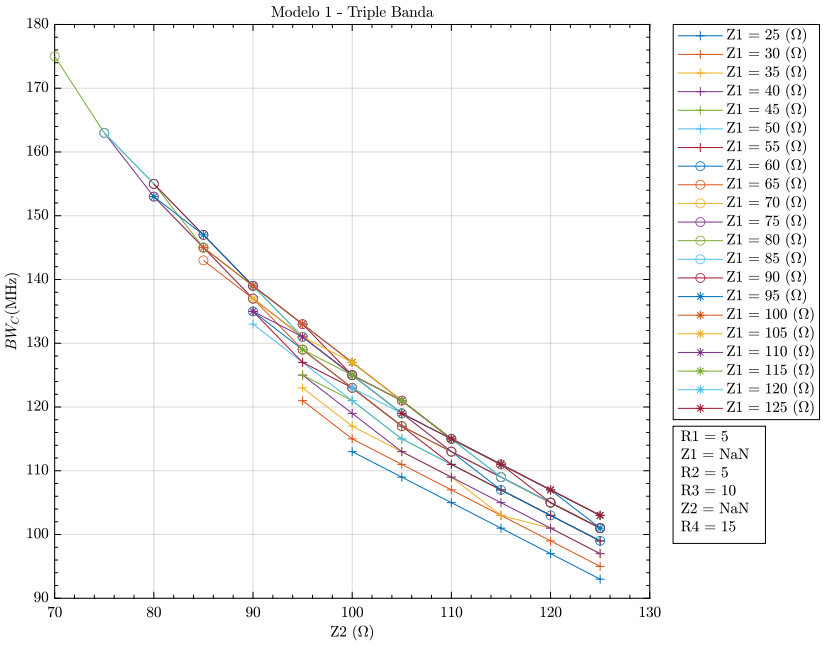


Figura 5.25: Curva de diseño en función del ancho de banda de la banda central

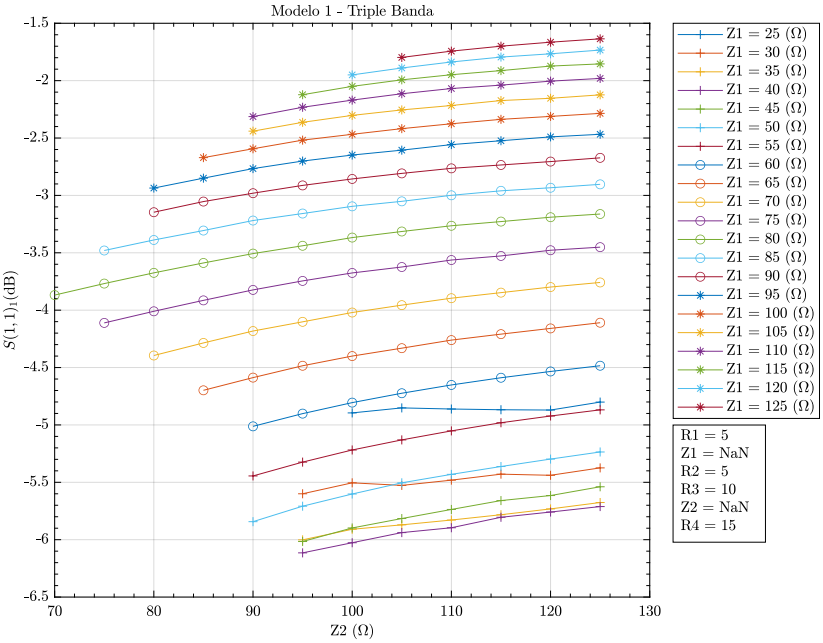


Figura 5.26: Curva de diseño en función del parámetro $S(1,1)$ de las bandas laterales

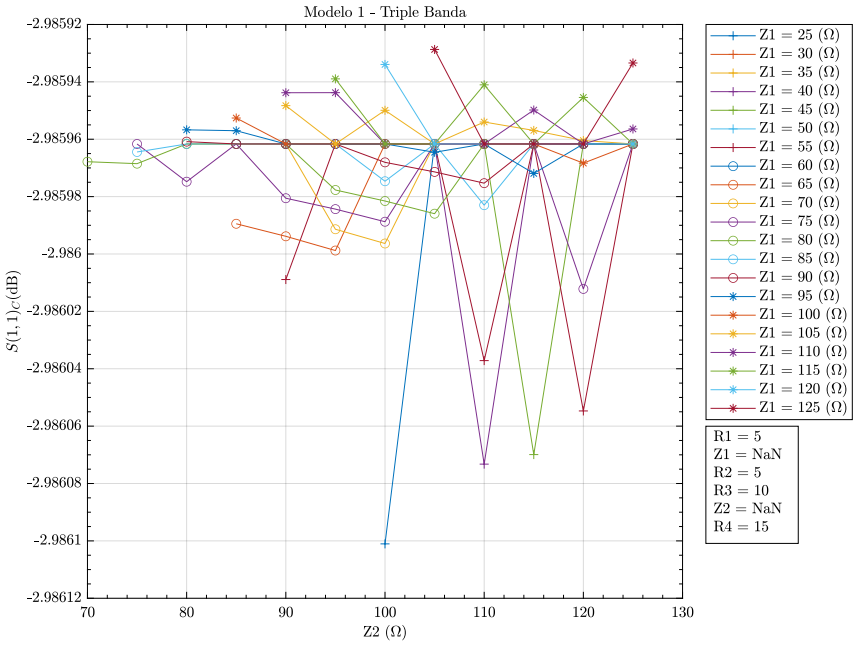


Figura 5.27: Curva de diseño en función del parámetro $S(1,1)$ de la banda central

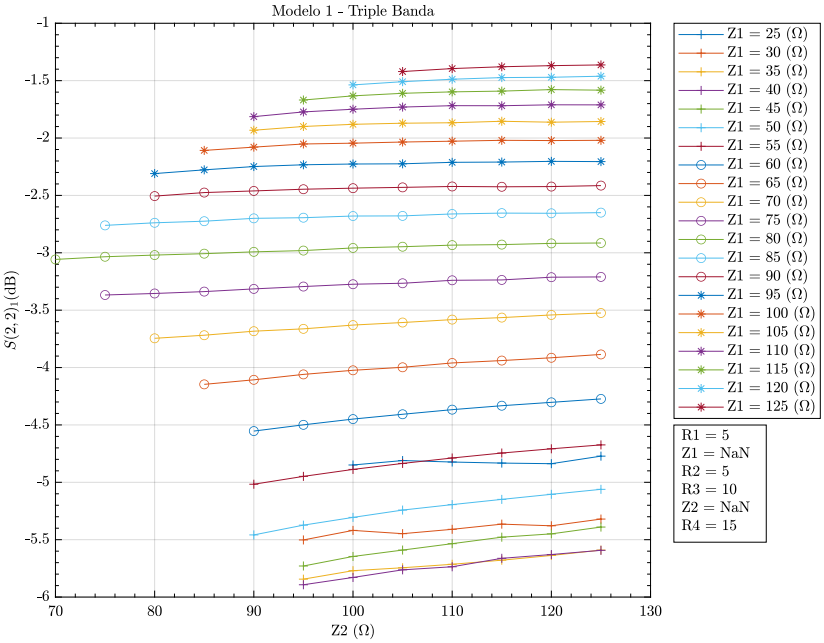


Figura 5.28: Curva de diseño en función del parámetro $S(2,2)$ de las bandas laterales

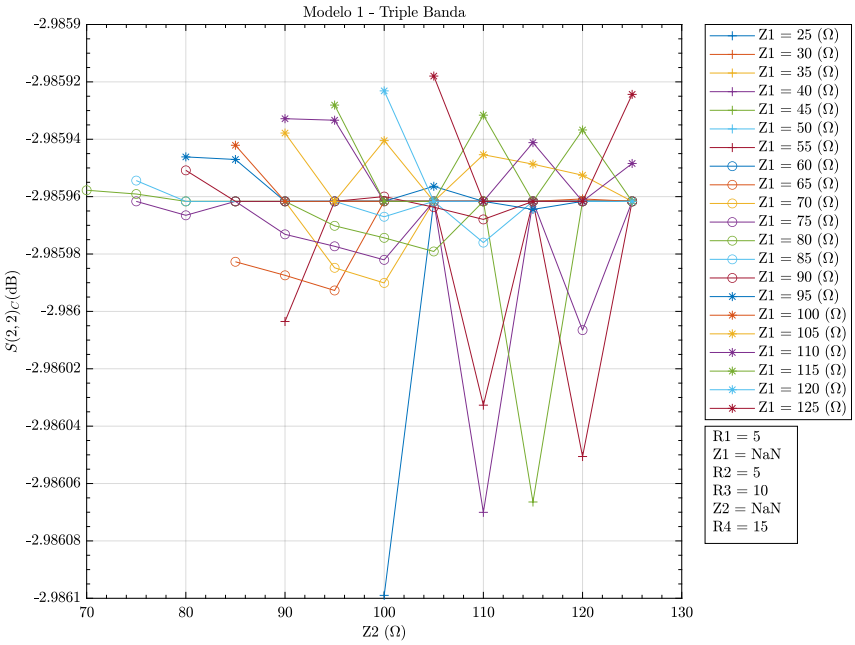


Figura 5.29: Curva de diseño en función del parámetro $S(2,2)$ de la banda central

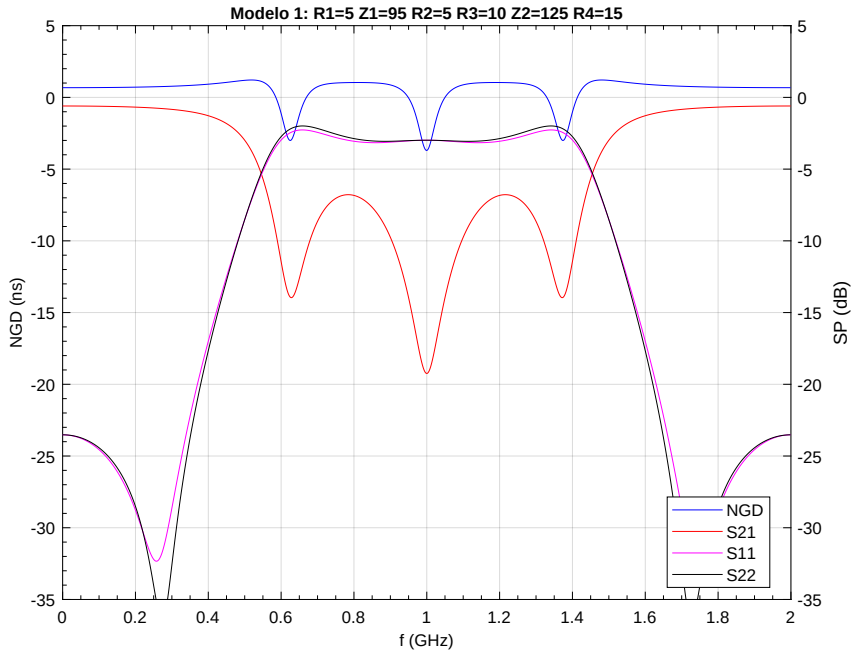


Figura 5.30: Respuesta ideal para la configuración seleccionada

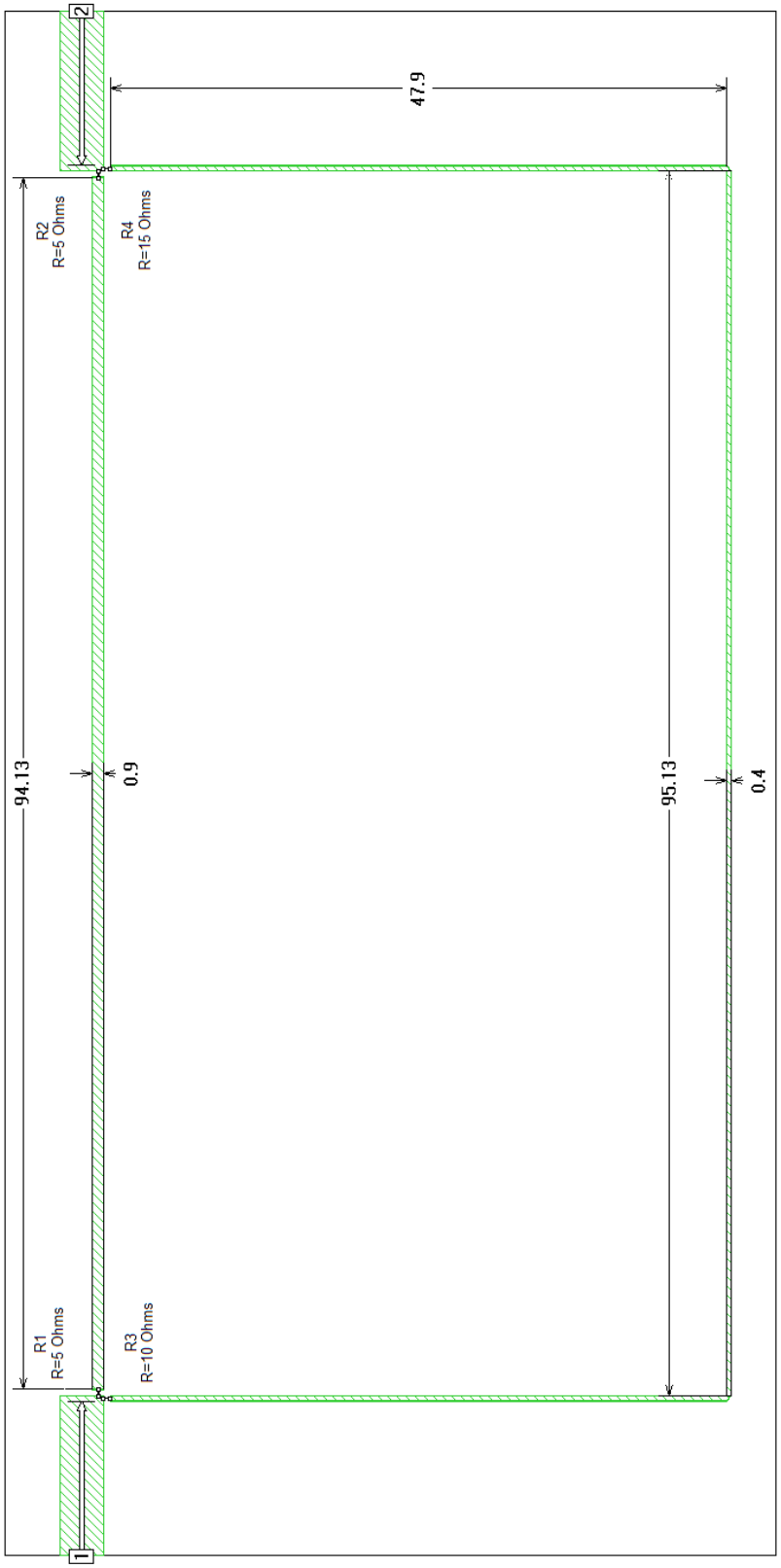


Figura 5.31: Diseño del prototipo en Sonnet (Rogers 4003)

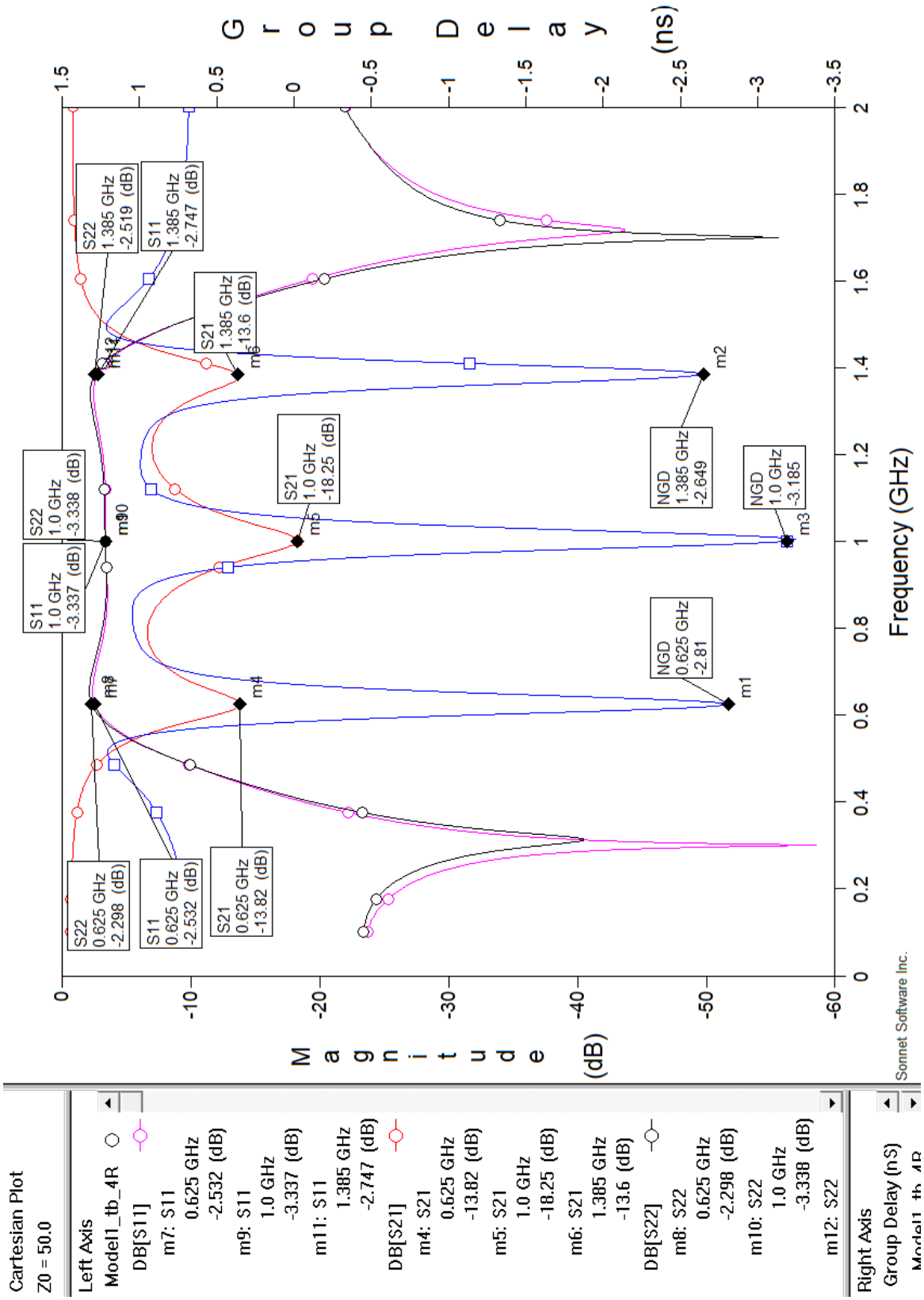


Figura 5.32: Respuesta del prototipo en Sonnet (Rogers 4003)

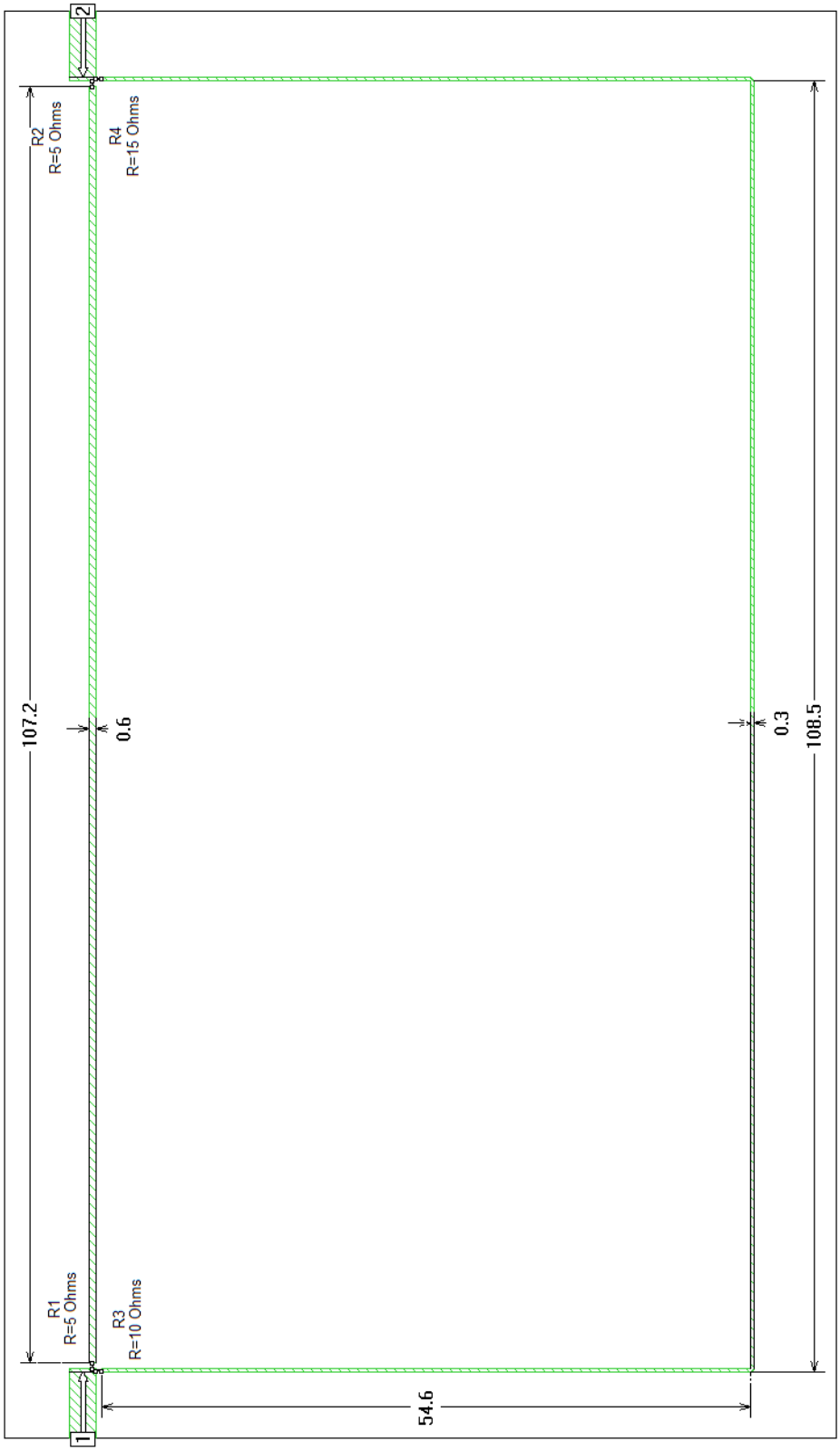


Figura 5.33: Diseño del prototipo en Sonnet (TLX-8)

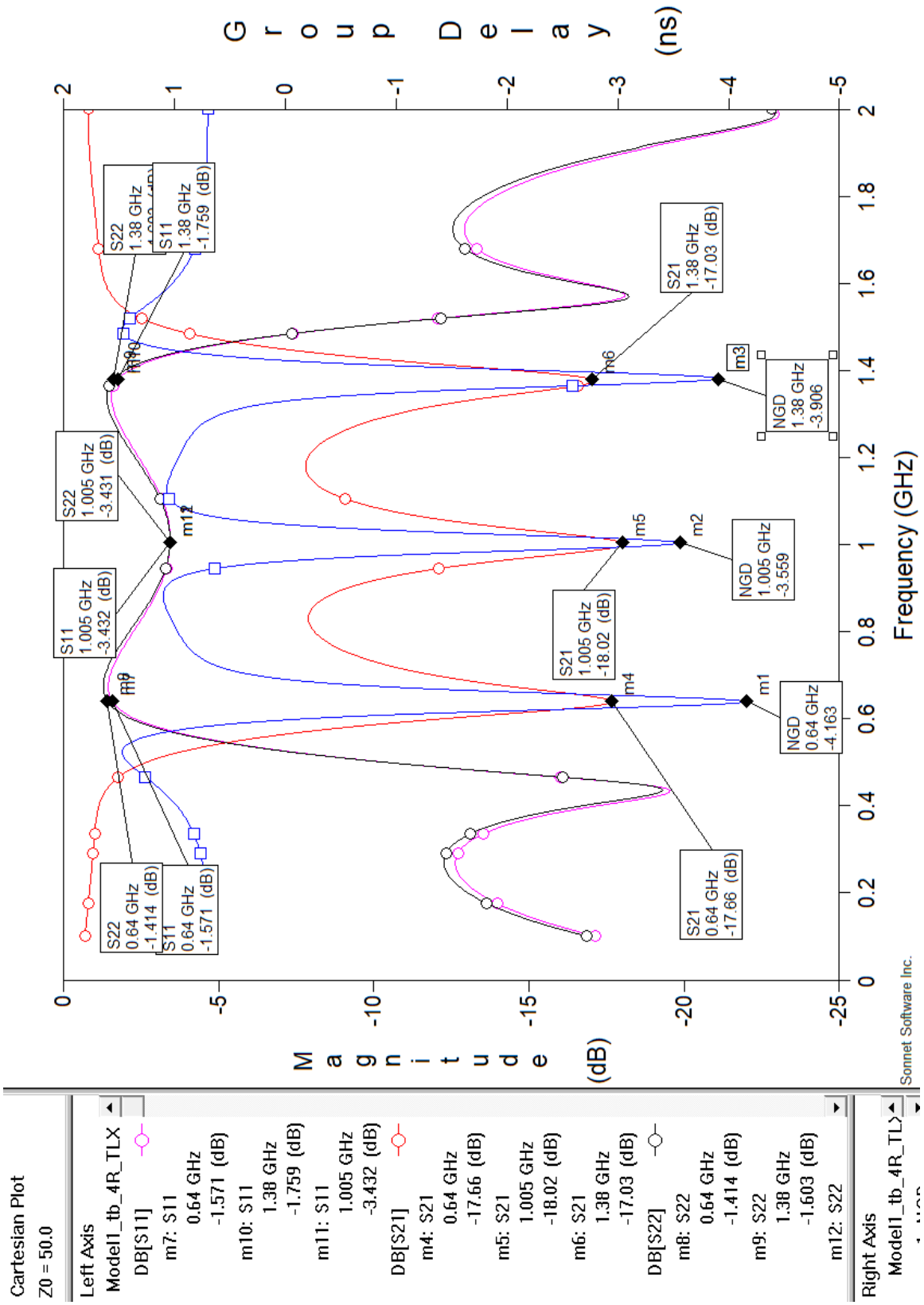


Figura 5.34: Respuesta del prototipo en Sonnet (TLX-8)

Finalmente, los valores obtenidos para todas las bandas tras la simulación de onda completa son más similares a los calculados idealmente que en el caso anterior y se muestran en la Tabla 5.10.

		NGD (ns)	f (MHz)	S_{21}(dB)	S_{11}(dB)	S_{22}(dB)	BW (MHz)
Banda 1	Ideal	-3,003	625	-13,92	-2,467	-2,205	85
	Rogers 4003	-2,81	625	-13,82	-2,532	-2,298	85
	TLX-8	-4,163	640	-17,66	-1,571	-1,414	90
Banda Central	Ideal	-3,697	1000	-19,24	-2,986	-2,986	101
	Rogers 4003	-3,241	1005	-18,21	-3,333	-3,334	95
	TLX-8	-3,559	1005	-18,02	-3,432	-3,431	80
Banda 2	Ideal	-3,003	1375	-13,92	-2,467	-2,205	85
	Rogers 4003	-2,649	1385	-13,6	-2,747	-2,519	85
	TLX-8	-3,906	1380	-17,03	-1,759	-1,603	90

Tabla 5.10: Comparación entre la respuesta ideal y la simulación en onda completa - Triple Banda

5.1.3. Modelo 1 - Banda Central

Ya por último restaría comentar el análisis de la topología para obtener una respuesta con una única banda central. Este era, tal y como se comentó en el Capítulo 3 el objetivo principal de este TFM. Sin embargo los resultados que se obtienen no son excesivamente interesantes, por lo que no se ha implementado ningún prototipo y simulado en onda completa. A continuación se exponen los valores mínimos y máximos para cada parámetro (Tabla 5.11 al igual que la distribución de las conbinaciones en la Figura 5.35. De éstos datos es posible extraer que no es posible obtener características adecuadas en cuanto a NGD, atenuación y ancho de banda.

Parámetro	Mínimo	Máximo
$\boldsymbol{NGD}$	$-3,4647$ ns	-1 ns
$\boldsymbol{S_{21}}$	-20 dB	$-10,9306$ dB
$\boldsymbol{S_{11}}$	-343 dB	$-1,2136$ dB
$\boldsymbol{S_{22}}$	-343 dB	$-1,2136$ dB
$\boldsymbol{BW}$	55 MHz	293 MHz
$\boldsymbol{BW_{1dB}}$	36 MHz	104 MHz
$\boldsymbol{BW_{3dB}}$	70 MHz	196 MHz

Tabla 5.11: Análisis numérico del Modelo 1 - Banda Central

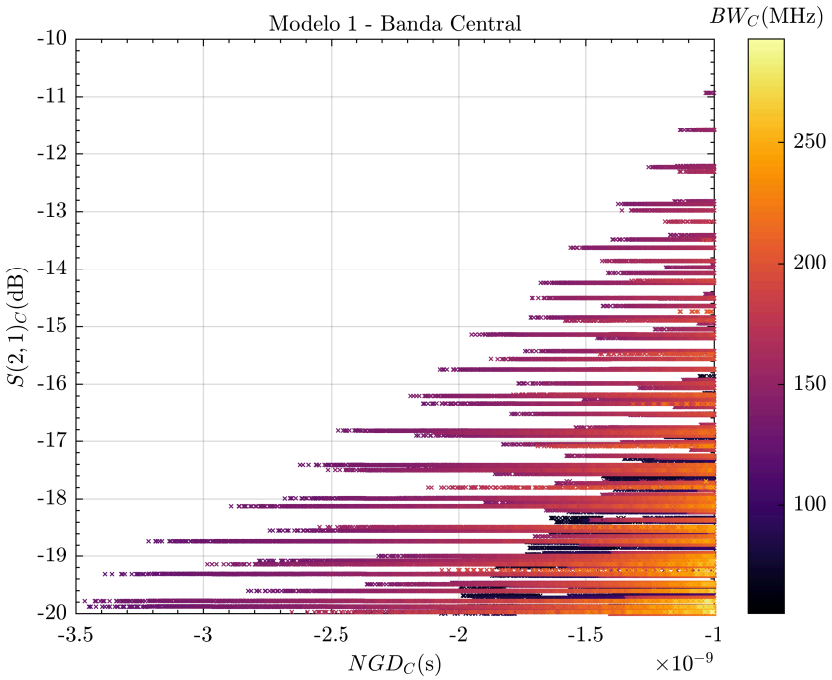


Figura 5.35: NGD frente a atenuación y ancho de banda (Análisis Modelo 1 - Banda Central)

6. Conclusiones y líneas futuras

El presente TFM tenía por objetivo realizar un estudio de estructuras con retardo de grupo negativo (NGD) en tecnología microstrip. Inicialmente pretendía evaluar la topología del Modelo 1, que es relativamente sencilla, para conseguir un ancho de banda, NGD y atenuación capaces de competir con otras estructuras publicadas, como las que se expusieron en el Capítulo 1. Sin embargo, las características de esta topología han supuesto una variación en la metodología planteada inicialmente: en este TFM no sólo se ha realizado un estudio de una topología concreta, sino que se ha desarrollado un sistema capaz de evaluar paramétricamente por fuerza bruta ésta u otras topologías similares.

La fuerza de computación disponible actualmente ha permitido que de un diseño del cual inicialmente no parecía poder extraerse grandes resultados, sea posible parametrizarlo en base a las variables de mayor interés. Tanto las curvas de diseño mostradas como los diseños simulados en onda completa no tienen por qué ser aquellos con mejores características, sino que son sólo una muestra de lo que el sistema de análisis es capaz de ofrecer: la posibilidad de encontrar rápidamente combinaciones que se adapten a los requisitos.

Una de las posibilidades que surgieron durante la realización de este trabajo, fue introducir una resistencia intermedia en las líneas de transmisión, para comprobar si así era posible anular los ceros de transmisión y conseguir mejores características. Esta variación da pie a tres posibles circuitos que se muestran en la Figura 6.1.

Puesto que el sistema de cómputo desarrollado en Python es escalable, fue posible introducir los cambios pertinentes en el código para computar el Modelo 2 y el Modelo 3. Para computar dichos modelos se utilizó Google Cloud Platform teniendo en cuenta que la variable T, que indica la longitud eléctrica de la línea en cuestión, varía en pasos de 10°. Además, puesto que el sistema es simétrico, no es necesario que la variable T_1 tome todos los valores, sino sólo aquellos de 0 a 90°, al igual que la variable T_2 sólo de 0 a 180°. Pese a estas reducciones, cada vez que se computaba el Modelo 2 o 3 para un valor de T fijo el tiempo necesario de cómputo era de 30 horas en una instancia de Google de 16 procesadores virtuales (a demanda) y 14,5 GB de memoria RAM; o de 35 horas en el ordenador personal utilizado para el cómputo del Modelo 1 de este TFM. Esto supone que son necesarias 300 horas de computación en Google Cloud Platform para el Modelo 2 y 600 horas para el Modelo 3.

Los Modelos 2 y 3 fueron computados y a día de hoy es posible analizarlos directamente con las herramientas desarrolladas en MATLAB ® o introduciendo algunos pequeños cambios para adaptar el nombre de las variables y el número de éstas. Sin

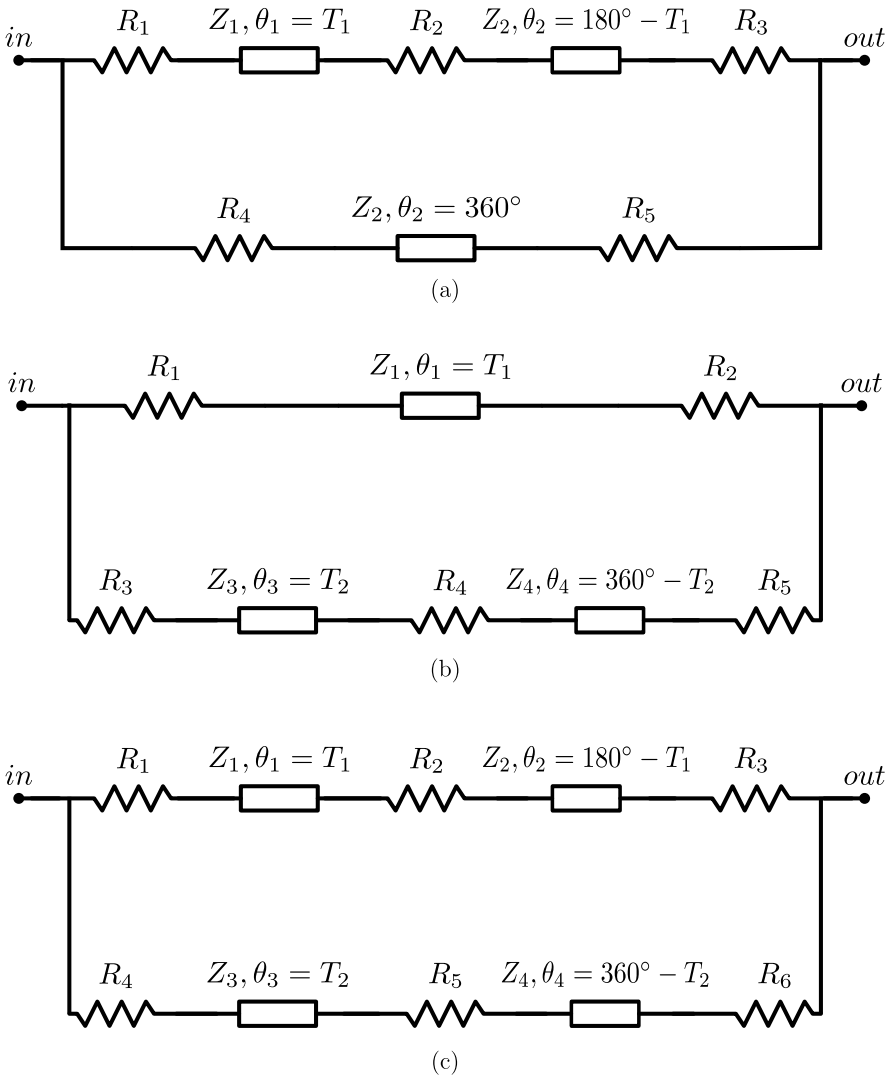


Figura 6.1: Variaciones del modelo propuesto. (a) Modelo 2. (b) Modelo 3. (c) Modelo 4.

embargo, estos resultados se escapan de los objetivos de este TFM y por ello se relatan en este capítulo. También a modo de línea futura, el sistema está preparado para poder computar el Modelo 4 y analizar los Modelos 2 y 3 para parametrizar sus respuestas.

Ya para terminar, este TFM no sólo ha cumplido su cometido en cuanto al aprendizaje de los conceptos principales de la ingeniería de microondas, sino que también ha sido un reto personal encontrar formas distintas a las tradicionales para realizar el análisis paramétrico.

Bibliografía

[1] D. M. Pozar, *Microwave engineering*, 4th ed. JohnWiley & Sons, Inc., 2012, ch. 1, pp. 1–2.

[2] "BOE: Orden ETU/1033/2017, de 25 de octubre, por la que se aprueba el cuadro nacional de atribuciones de frecuencias," p. 364, 2017. [Online]. Available: https://www.boe.es/eli/es/o/2017/10/25/etu1033

[3] "IEEE Standard 802.11 for Wireless LAN," Tech. Rep., Dec. 2016. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7786995

[4] G. Chaudhary, Y. Jeong, and J. Lim, "Microstrip Line Negative Group Delay Filters for Microwave Circuits," *IEEE Transactions on Microwave Theory and Techniques*, vol. 62, no. 2, pp. 234–243, Feb. 2014.

[5] A. Sommerfeld, *Lectures on theoretical physics: Optics*. Academic press, 1954, vol. 4.

[6] L. Brillouin, *Wave propagation and group velocity*. Academic Press, 1960, vol. 8.

[7] B. Ravelo, "Investigation on Microwave Negative Group Delay Circuit," *Electromagnetics*, vol. 31, no. 8, pp. 537–549, Nov. 2011. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/02726343.2011.621106

[8] C. G. B. Garrett and D. E. McCumber, "Propagation of a Gaussian Light Pulse through an Anomalous Dispersion Medium," *Physical Review A*, vol. 1, no. 2, pp. 305–313, Feb. 1970. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.1.305

[9] B. Segard and B. Macke, "Observation of negative velocity pulse propagation," *Physics Letters A*, vol. 109, no. 5, pp. 213–216, May 1985. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/0375960185903056

[10] B. Macke and B. Ségard, "Propagation of light-pulses at a negative group-velocity," *The European Physical Journal D - Atomic, Molecular and Optical Physics*, vol. 23, no. 1, pp. 125–141, Apr. 2003. [Online]. Available: http://www.springerlink.com/Index/10.1140/epjd/e2003-00022-0

[11] K. T. McDonald, "Negative group velocity," *American Journal of Physics*, vol. 69, no. 5, pp. 607–614, 2001.

[12] O. F. Siddiqui, M. Mojahedi, and G. V. Eleftheriades, "Periodically loaded transmission line with effective negative refractive index and negative group velocity," *IEEE Transactions on Antennas and Propagation*, vol. 51, no. 10, pp. 2619–2625, Oct. 2003.

[13] J. F. Woodley and M. Mojahedi, "Negative group velocity and group delay in left-handed media," *Phys. Rev. E*, vol. 70, no. 4, p. 046603, Oct. 2004. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.70.046603

[14] E. L. Bolda, R. Y. Chiao, and J. C. Garrison, "Two theorems for the group velocity in dispersive media," *Physical Review A*, vol. 48, no. 5, pp. 3890–3894, Nov. 1993. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.48.3890

[15] M. W. Mitchell and R. Y. Chiao, "Negative group delay and "fronts" in a causal system: An experiment with very low frequency bandpass amplifiers," *Physics Letters A*, vol. 230, no. 3-4, pp. 133–138, Jun. 1997. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0375960197002442

[16] M. A. Sanchez-Soriano, J. Dura, S. Sirci, and S. Marini, "Signal-Interference-Based Structure with Negative Group Delay Properties," in *2018 48th European Microwave Conference (EuMC)*. Madrid: IEEE, Sep. 2018, pp. 1021–1024. [Online]. Available: https://ieeexplore.ieee.org/document/8541678/

[17] R. Y. Chiao, E. L. Bolda, J. Bowie, J. Boyce, and M. W. Mitchell, "Superluminality and amplifiers," *Progress in Crystal Growth and Characterization of Materials*, vol. 33, no. 1, pp. 319 – 325, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0960897496836631

[18] C. M. Wu and T. Itoh, "Maximally Flat Negative Group-Delay Circuit: A Microwave Transversal Filter Approach," *IEEE Transactions on Microwave Theory and Techniques*, vol. 62, no. 6, pp. 1330–1342, Jun. 2014.

[19] M. Kandic and G. E. Bridges, "Bilateral Gain-Compensated Negative Group Delay Circuit," *IEEE Microwave and Wireless Components Letters*, vol. 21, no. 6, pp. 308–310, Jun. 2011.

[20] S. Lucyszyn, I. Robertson, and A. Aghvami, "Negative group delay synthesiser," *Electronics Letters*, vol. 29, no. 9, p. 798, 1993. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/el_19930533

[21] S. Lucyszyn and I. D. Robertson, "Analog reflection topology building blocks for adaptive microwave signal processing applications," *IEEE Transactions on Microwave Theory and Techniques*, vol. 43, no. 3, pp. 601–611, 1995.

[22] B. Ravelo, A. Pérennec, M. Le Roy, and Y. G. Boucher, "Active microwave circuit with negative group delay," *IEEE Microwave and Wireless Components Letters*, vol. 17, no. 12, pp. 861–863, 2007.

[23] B. Ravelo, A. Pérennec, and M. Le Roy, "Synthesis of broadband negative group delay active circuits," in *2007 IEEE/MTT-S International Microwave Symposium*. IEEE, 2007, pp. 2177–2180.

[24] M. Kandic and G. E. Bridges, "Asymptotic limits of negative group delay in active resonator-based distributed circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 8, pp. 1727–1735, 2011.

[25] P. Mondal and M. Mandal, "Design of sharp-rejection, compact, wideband bandstop filters," *IET Microwaves, Antennas & Propagation*, vol. 2, no. 4, pp. 389–393, Jun. 2008. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/iet-map_20070212

[26] R. Gomez-Garcia and J. I. Alonso, "Design of sharp-rejection and low-loss wide-band planar filters using signal-interference techniques," *IEEE microwave and wireless components letters*, vol. 15, no. 8, pp. 530–532, 2005.

[27] M. Á. Sánchez-Soriano, G. Torregrosa-Penalva, and E. Bronchalo, "Compact wideband bandstop filter with four transmission zeros," *IEEE Microwave and Wireless Components Letters*, vol. 20, no. 6, pp. 313–315, 2010.

[28] H. Choi, Y. Jeong, C. D. Kim, and J. S. Kenney, "Efficiency enhancement of feedforward amplifiers by employing a negative group-delay circuit," *IEEE Transactions on Microwave Theory and Techniques*, vol. 58, no. 5, pp. 1116–1125, 2010.

[29] C. Broomfield and J. Everard, "Broadband negative group delay networks for compensation of microwave oscillators and filters," *Electronics Letters*, vol. 36, no. 23, pp. 1931–1933, 2000.

[30] G. Chaudhary, Y. Jeong, and J. Lim, "Realization of Negative Group Delay Network Using Defected Microstrip Structure," *International Journal of Antennas and Propagation*, 2014. [Online]. Available: https://www.hindawi.com/journals/ijap/2014/836960/

[31] G. Chaudhary and Y. Jeong, "A design of compact wideband negative group delay network using cross coupling," *Microwave and Optical Technology Letters*, vol. 56, no. 11, pp. 2495–2497, 2014. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/mop.28627

[32] H. Choi, S. Shim, Y. Jeong, J. Lim, C. D. Kim, S. Eom, and Y. Jung, "2.14/3.5 GHz novel dual-band negative group delay circuit design based on composite right/left handed transmission line," in *The 40th European Microwave Conference*, Sep. 2010, pp. 441–444.

[33] B. Ravelo and S. De Blasi, "An FET-based microwave active circuit with dual-band negative group delay," *Journal of Microwaves, Optoelectronics and Electromagnetic Applications*, vol. 10, no. 2, pp. 355–366, 2011.

[34] H. Choi, Y. Jeong, J. Lim, S. Eom, and Y. Jung, "A Novel Design for a Dual-Band Negative Group Delay Circuit," *IEEE Microwave and Wireless Components Letters*, vol. 21, no. 1, pp. 19–21, Jan. 2011.

[35] G. Chaudhary, Y. Jeong, and J. Lim, "Miniaturized Dual-Band Negative Group Delay Circuit Using Dual-Plane Defected Structures," *IEEE Microwave and Wireless Components Letters*, vol. 24, no. 8, pp. 521–523, Aug. 2014.

[36] G. Chaudhary, P. Kim, J. Jeong, Y. Jeong, and J. Lim, "Dual-band negative group delay circuit using defected microstrip structure," in *2015 IEEE Radio and Wireless Symposium (RWS)*, Jan. 2015, pp. 129–131.

[37] H. Taher and R. Farrell, "Dual wide-band miniaturized negative group delay circuit using open circuit stubs," *Microwave and Optical Technology Letters*, vol. 60, no. 2, pp. 428–432, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/mop.30979

[38] T. Shao, S. Fang, Z. Wang, and H. Liu, "A Compact Dual-Band Negative Group Delay Microwave Circuit," *Radioengineering*, vol. 27, no. 4, pp. 1070–1076, Dec. 2018.

[39] D. M. Pozar, *Microwave engineering*, 4th ed. JohnWiley & Sons, Inc., 2012, ch. 2.

[40] D. Grieg and H. Engelmann, "Microstrip-a new transmission technique for the klilomegacycle range," *Proceedings of the IRE*, vol. 40, no. 12, pp. 1644–1650, 1952.

[41] R. M. Barrett, "Microwave printed circuits - the early years," *IEEE Transactions on Microwave Theory and Techniques*, vol. 32, no. 9, pp. 983–990, 1984.

[42] D. M. Pozar, *Microwave engineering*, 4th ed. JohnWiley & Sons, Inc., 2012, ch. 3.

[43] R. E. Collin, *Foundations for microwave engineering*, 2nd ed. John Wiley & Sons, 2001, ch. 3.

[44] I. J. Bahl, "A designer's guide to microstrip line." 1977.

[45] D. M. Pozar, *Microwave engineering*, 4th ed. JohnWiley & Sons, Inc., 2012, ch. 4.

[46] J. M. R. Plens, "Setfig paper," https://github.com/jmrplens/SetFigPaper, 2020.

A. Equivalencias entre parámetros para redes de dos puertos

	S	Z	Y	$ABCD$
S_{11}	S_{11}	$\frac{(Z_{11}-Z_0)(Z_{22}+Z_0)-Z_{12}Z_{21}}{\Delta Z}$	$\frac{(Y_0-Y_{11})(Y_0+Y_{22})+Y_{12}Y_{21}}{\Delta Y}$	$\frac{A+B/Z_0-CZ_0-D}{A+B/Z_0+CZ_0+D}$
S_{12}	S_{12}	$\frac{2Z_{12}Z_0}{\Delta Z}$	$\frac{-2Y_{12}Y_0}{\Delta Y}$	$\frac{2(AD-BC)}{A+B/Z_0+CZ_0+D}$
S_{21}	S_{21}	$\frac{2Z_{21}Z_0}{\Delta Z}$	$\frac{-2Y_{21}Y_0}{\Delta Y}$	$\frac{2}{A+B/Z_0+CZ_0+D}$
S_{22}	S_{22}	$\frac{(Z_{11}+Z_0)(Z_{22}-Z_0)-Z_{12}Z_{21}}{\Delta Z}$	$\frac{(Y_0+Y_{11})(Y_0-Y_{22})+Y_{12}Y_{21}}{\Delta Y}$	$\frac{-A+B/Z_0-CZ_0+D}{A+B/Z_0+CZ_0+D}$
Z_{11}	$Z_0\frac{(1+S_{11})(1-S_{22})+S_{12}S_{21}}{(1-S_{11})(1-S_{22})-S_{12}S_{21}}$	Z_{11}	$\frac{Y_{22}}{\lvert Y\rvert}$	$\frac{A}{C}$
Z_{12}	$Z_0\frac{2S_{12}}{(1-S_{11})(1-S_{22})-S_{12}S_{21}}$	Z_{12}	$\frac{-Y_{12}}{\lvert Y\rvert}$	$\frac{AD-BC}{C}$
Z_{21}	$Z_0\frac{2S_{21}}{(1-S_{11})(1-S_{22})-S_{12}S_{21}}$	Z_{21}	$\frac{-Y_{21}}{\lvert Y\rvert}$	$\frac{1}{C}$
Z_{22}	$Z_0\frac{(1-S_{11})(1+S_{22})+S_{12}S_{21}}{(1-S_{11})(1-S_{22})-S_{12}S_{21}}$	Z_{22}	$\frac{Y_{11}}{\lvert Y\rvert}$	$\frac{D}{C}$
Y_{11}	$Y_0\frac{(1-S_{11})(1+S_{22})+S_{12}S_{21}}{(1+S_{11})(1+S_{22})-S_{12}S_{21}}$	$\frac{Z_{22}}{\lvert Z\rvert}$	Y_{11}	$\frac{D}{B}$
Y_{12}	$Y_0\frac{-2S_{12}}{(1+S_{11})(1+S_{22})-S_{12}S_{21}}$	$\frac{-Z_{12}}{\lvert Z\rvert}$	Y_{12}	$\frac{BC-AD}{B}$
Y_{21}	$Y_0\frac{-2S_{21}}{(1+S_{11})(1+S_{22})-S_{12}S_{21}}$	$\frac{-Z_{21}}{\lvert Z\rvert}$	Y_{21}	$\frac{-1}{B}$
Y_{22}	$Y_0\frac{(1+S_{11})(1-S_{22})+S_{12}S_{21}}{(1+S_{11})(1+S_{22})-S_{12}S_{21}}$	$\frac{Z_{11}}{\lvert Z\rvert}$	Y_{22}	$\frac{A}{B}$
A	$\frac{(1+S_{11})(1-S_{22})+S_{12}S_{21}}{2S_{21}}$	$\frac{Z_{11}}{Z_{21}}$	$\frac{-Y_{22}}{Y_{21}}$	A
B	$Z_0\frac{(1+S_{11})(1+S_{22})-S_{12}S_{21}}{2S_{21}}$	$\frac{\lvert Z\rvert}{Z_{21}}$	$\frac{-1}{Y_{21}}$	B
C	$\frac{1}{Z_0}\frac{(1-S_{11})(1-S_{22})-S_{12}S_{21}}{2S_{21}}$	$\frac{1}{Z_{21}}$	$\frac{-\lvert Y\rvert}{Y_{21}}$	C
D	$\frac{(1-S_{11})(1+S_{22})+S_{12}S_{21}}{2S_{21}}$	$\frac{Z_{22}}{Z_{21}}$	$\frac{-Y_{11}}{Y_{21}}$	D
$\lvert Z\rvert = Z_{11}Z_{22};\quad \Delta Y = (Y_{11}+Y_0)(Y_{22}+Y_0)-Y_{12}Y_{21};\quad \Delta Z = (Z_{11}+Z_0)(Z_{22}+Z_0)-Z_{12}Z_{21};\quad Y_0 = 1/Z_0$				

Tabla A.1: Conversión entre parámetros de redes de dos puertos

B. Tutorial ADS

ADS es una herramienta de diseño y simulación de circuitos electrónicos desarrollada por Keysight[1]. Mediante este software es posible simular la respuesta de circuitos electrónicos para frecuencias de microondas. A continuación se muestra cómo generar los circuitos de este TFM.

B.1. Creación de un circuito con ADS

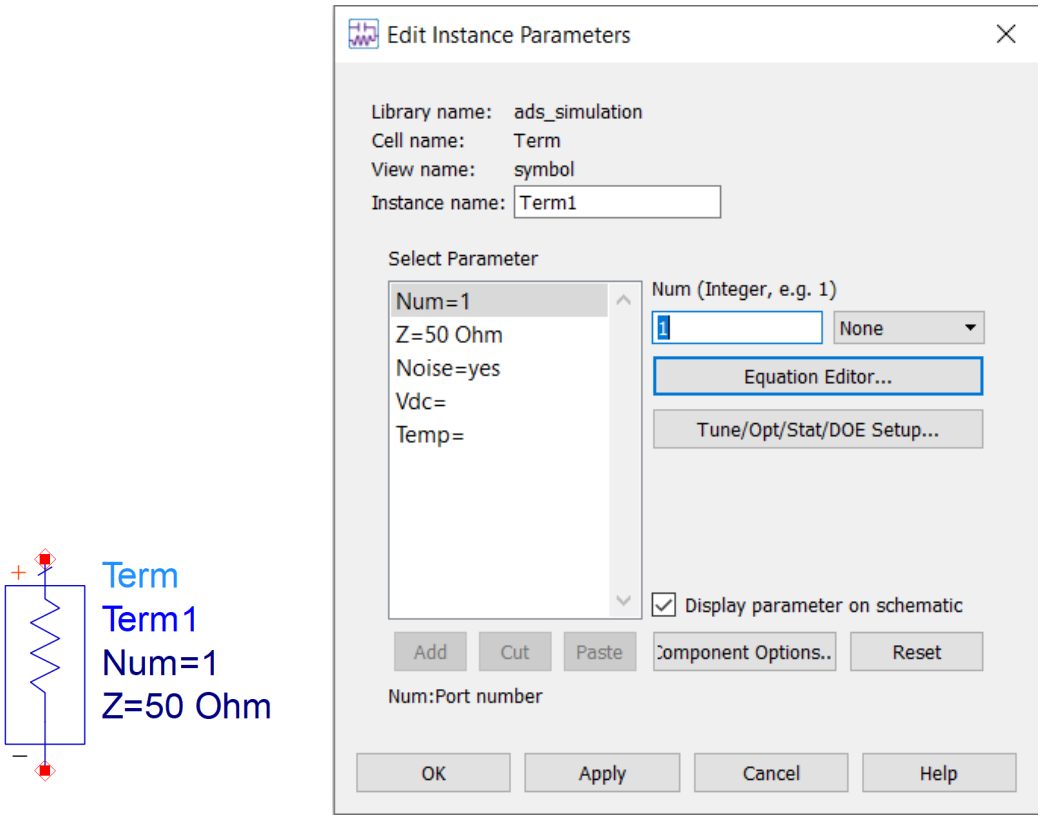
Tras configurar el espacio de trabajo, podemos crear un circuito nuevo (File →New →Schematic). Los componentes a utilizar para la creación de los circuitos de este TFM son:

[1]https://www.keysight.com/

B.1.1. Componentes genéricos

B.1.1.1. Term

El componente **Term** simboliza la impedancia característica de un puerto para simulaciones de parámetros S. En la Figura B.1 se muestra el símbolo y la configuración de sus propiedades. Este componente se encuentra en la paleta Simulation-S_Parameters.

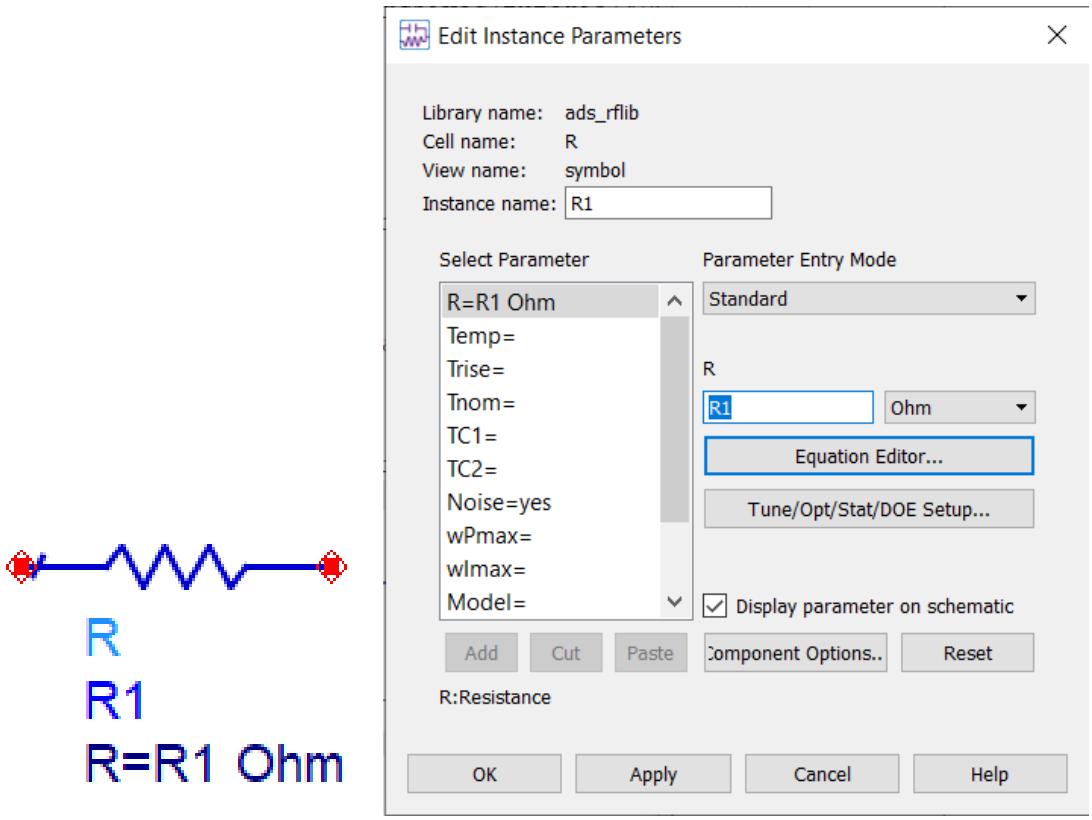


(a) Símbolo **(b)** Propiedades

Figura B.1: Componente Term

B.1.1.2. Resistencia

El componente **R** simboliza una resistencia. En la Figura B.2 se muestra el símbolo y la configuración de sus propiedades. Este componente se encuentra en la paleta Lumped-Components. Cabe destacar que en las propiedades (Figure B.2b) se indica como valor de resistencia el nombre de una variable (R1), que comentaremos más adelante

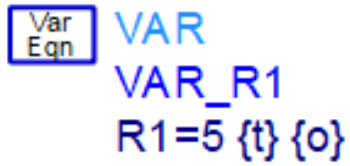


(a) Símbolo **(b)** Propiedades

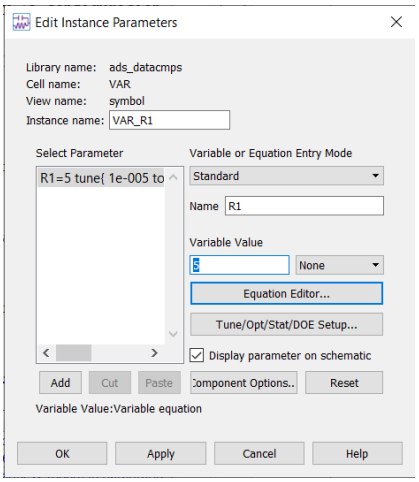
Figura B.2: Componente Resistencia

B.1.1.3. Variable

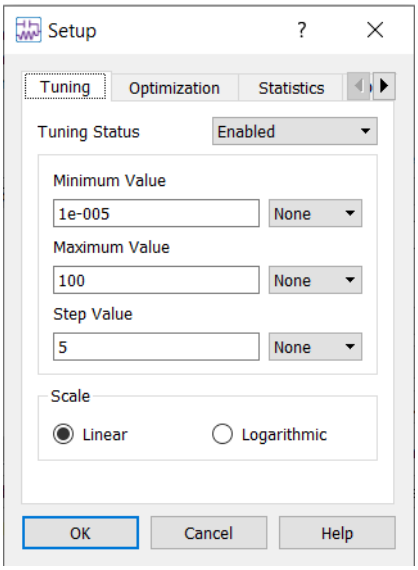
El componente **VAR** simboliza una resistencia. En la Figura B.3 se muestra el símbolo y la configuración de sus propiedades. Este componente se encuentra en la paleta Data Items. La configuración de Tunning y de Optimización no son obligatorias si no se van a realizar este tipo de simulaciones.



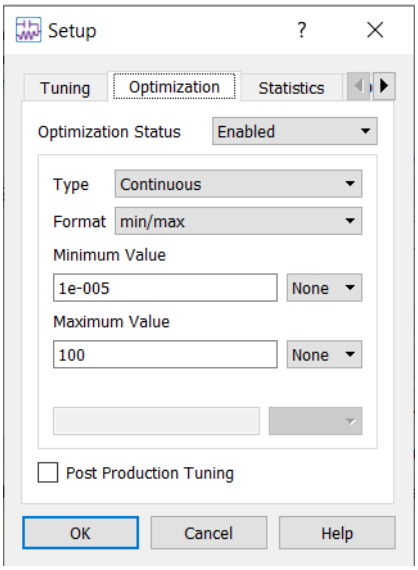
(a) Símbolo



(b) Propiedades



(c) Tuning (resistencia)

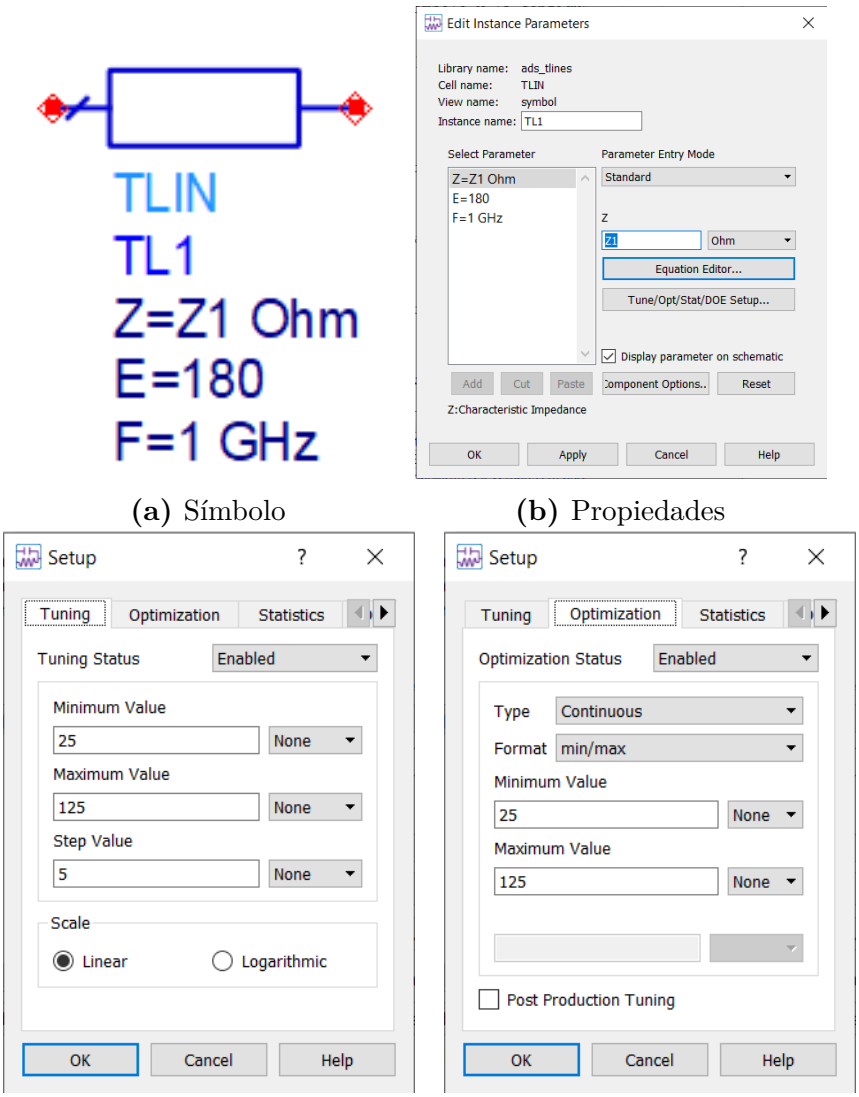


(d) Optimización (resistencia)

Figura B.3: Componente Variable

B.1.1.4. Línea de transmisión ideal

El componente **TLIN** simboliza una línea de transmisión ideal. En la Figura B.4 se muestra el símbolo y la configuración de sus propiedades. Este componente se encuentra en la paleta TLine-Ideal. En las Figuras B.4c y B.4d se muestran las configuraciones adecuadas para una variable asociada si se desea simular en modo Tunning u Optimización.



(a) Símbolo **(b)** Propiedades

(c) Tuning **(d)** Optimización

Figura B.4: Componente Variable

B.1.1.5. Línea de transmisión Microstrip

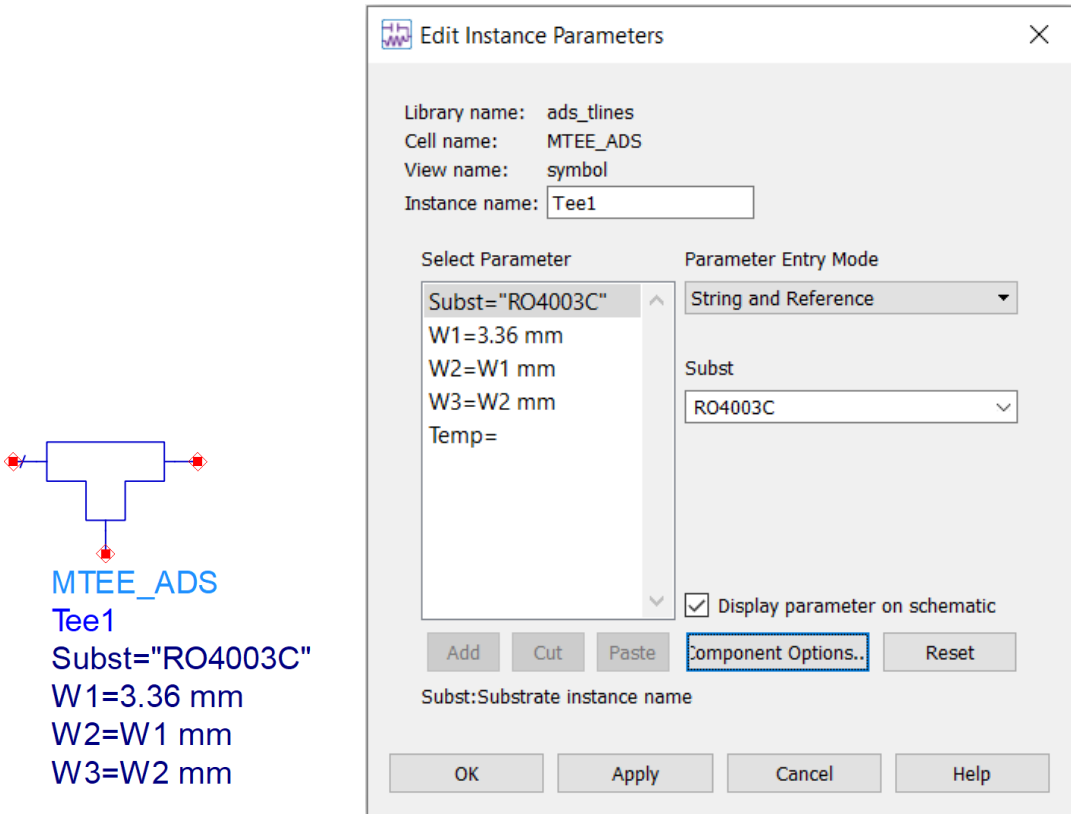
El componente **MLIN** simboliza una línea microstrip. En la Figura B.5 se muestra el símbolo y la configuración de sus propiedades. Este componente se encuentra en la paleta TLines-Microstrip. A diferencia de la línea ideal, en este caso se configuran las dimensiones físicas W y L, así como el substrato.

(a) Símbolo **(b)** Propiedades

Figura B.5: Componente para conexión en 'T'

B.1.1.6. Conexión microstrip en 'T'

El componente **MTEE** simboliza una conexión en 'T'. En la Figura B.6 se muestra el símbolo y la configuración de sus propiedades. Este componente se encuentra en la paleta TLines-Microstrip.

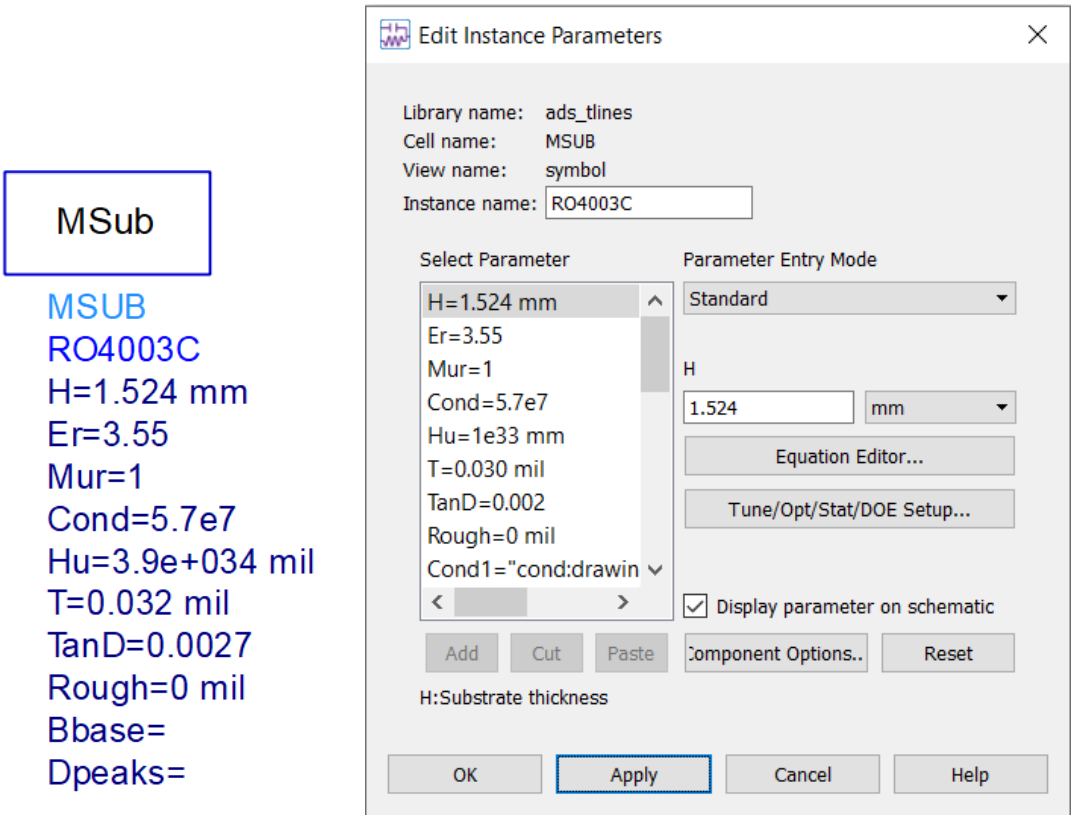


(a) Símbolo **(b)** Propiedades

Figura B.6: Componente conexión en 'T'

B.1.1.7. Substrato de componentes Microstrip

El componente **MSUB** define las propiedades del substrato utilizado para los componentes microstrip como los anteriores. En la Figura B.7 se muestra el símbolo y la configuración de sus propiedades. Este componente se encuentra en la paleta TLines-Microstrip.



(a) Símbolo **(b)** Propiedades

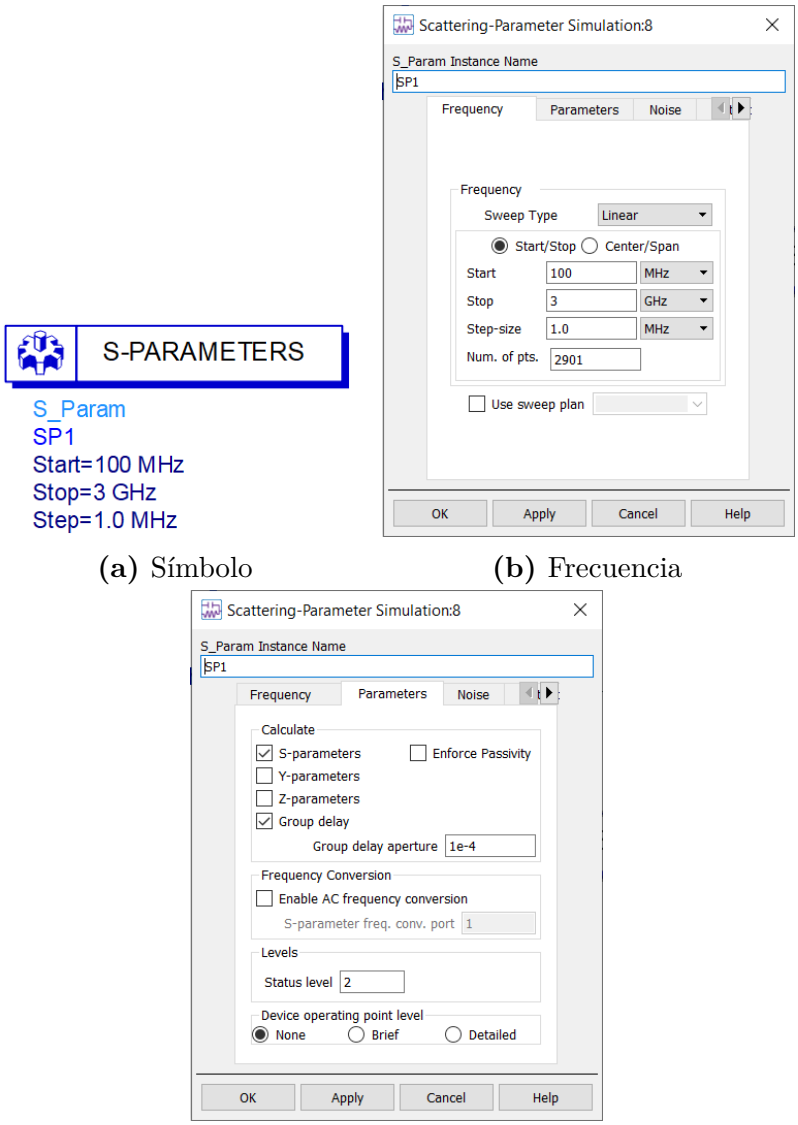
Figura B.7: Componente Substrato Microstrip

B.1.2. Componentes para Simulación

Los siguientes componentes son necesarios para simular el circuito, por lo que son más bien parámetros de configuración que componentes al uso.

B.1.2.1. Parámetros S

Necesario para indicar los parámetros a computar cuando se lance la simulación. En la Figura B.8 se muestra la configuración necesaria para poder computar también el retardo de grupo.



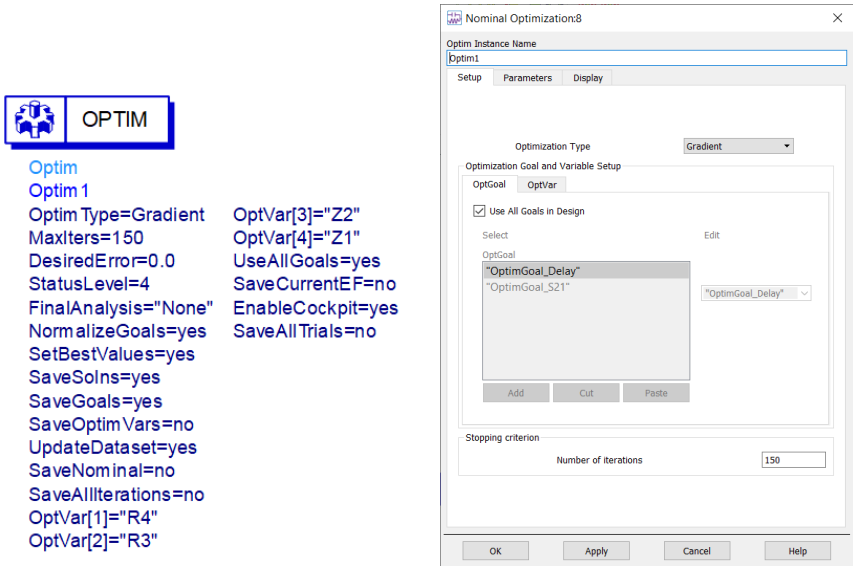
(a) Símbolo **(b)** Frecuencia

(c) Parámetros

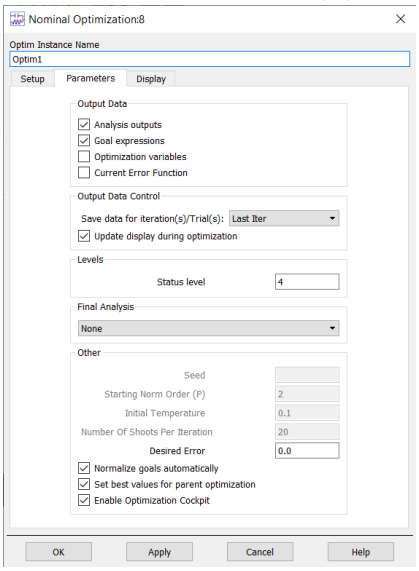
Figura B.8: Componente Parámetros S

B.1.2.2. Optimización

Este componente es necesario para indicar al software cómo debe realizar la optimización para alcanzar los objetivos. En la Figura B.8 se muestra la configuración necesaria para poder computar este tipo de simulación.



(a) Símbolo **(b)** Configuración

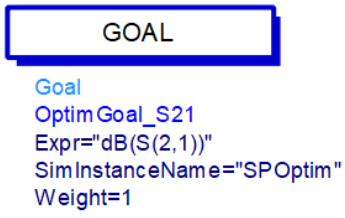


(c) Parámetros

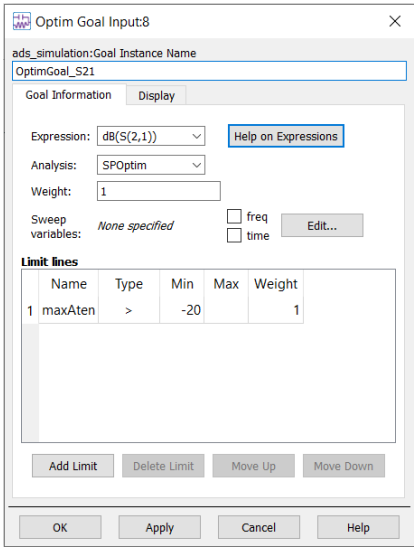
Figura B.9: Componente Optimización

B.1.2.3. Objetivo

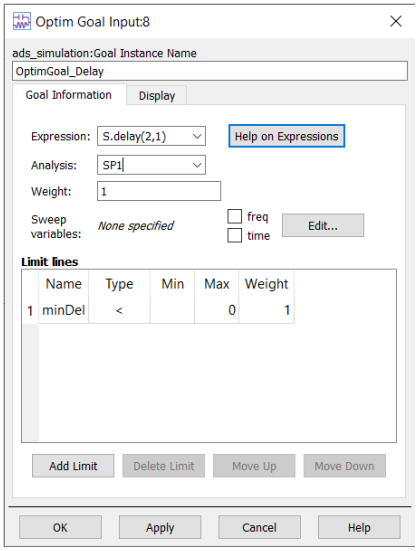
Este componente es necesario para indicar cual es el objetivo de la optimización. En la Figura B.10 se muestra la configuración necesaria para poder computar este tipo de simulación.



(a) Símbolo



(b) Objetivos Atenuación



(c) Objetivos NGD

Figura B.10: Componente Objetivos

B.1.3. Simulación y Visualización

Tras la construcción del circuito deseado, existen principalmente tres formas de lanzar la simulación: Simulación (estática), Tunning y Optimización. Todas ellas se encuentran en el menú *Simulate* de la barra de tareas. La diferencia entre la Simulación estática y una simulación con Tunning está en el valor que adquieren las variables. Mientras que para la primera se realiza una simulación con los valores que tengan en dicho momento, si se selecciona la segunda al lanzar la simulación se abre una nueva ventana (Figura B.11)que permite modificarlos. Esto es especialmente interesante de cara a realizar ajustes rápidos ya que se vuelve a calcular la respuesta al circuito tras cada modificación.

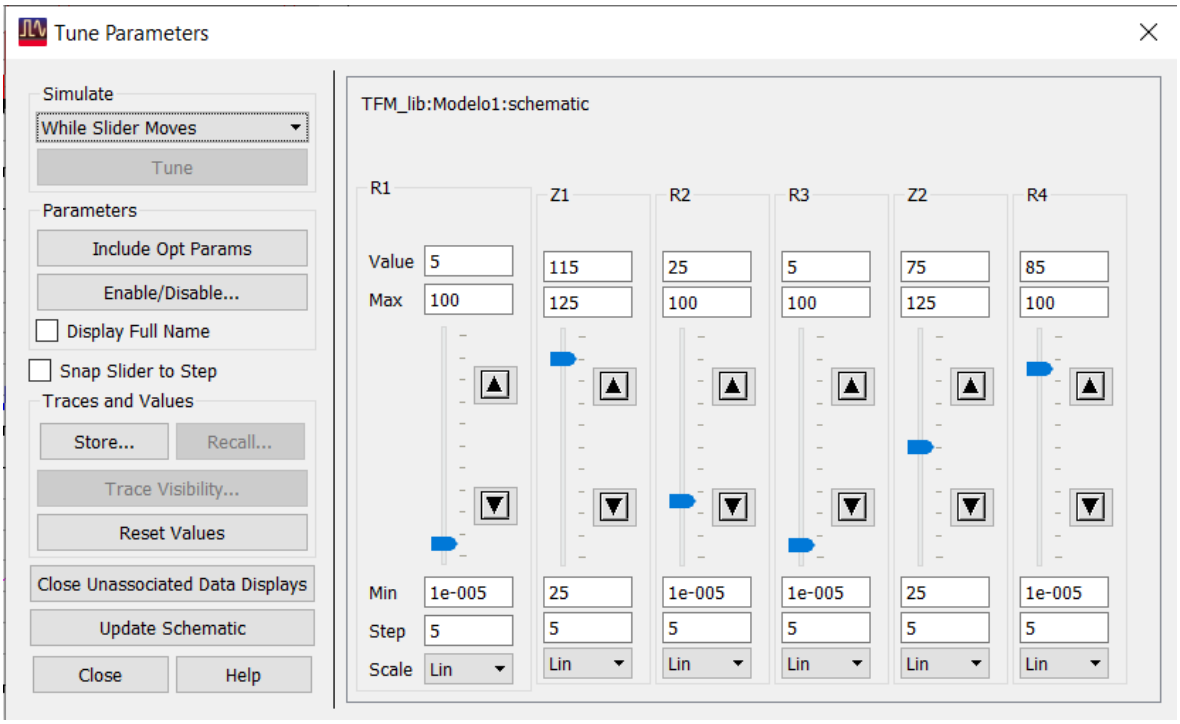
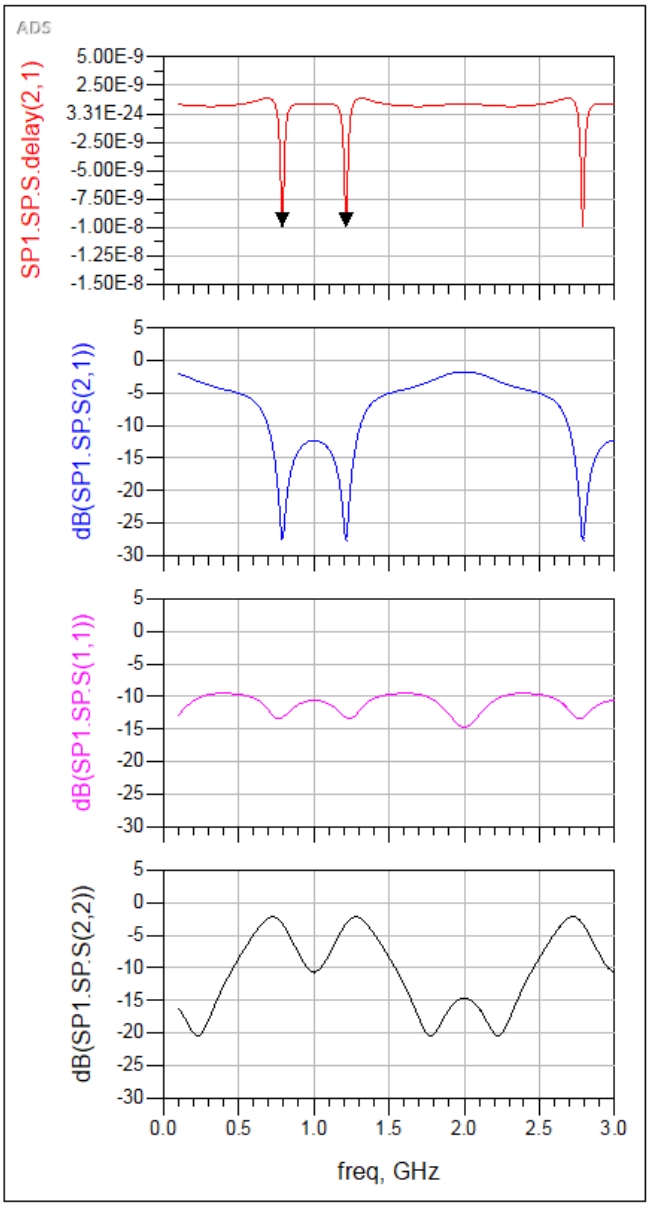


Figura B.11: Ventana de ajuste

Si se opta por optimizar las variables del circuito, tras realizar los cálculos pertinentes el programa ofrecerá la posibilidad de aplicar dichos valores a las variables.

En cualquiera de los casos, tras la simulación se abrirá una nueva ventana donde será posible incluir las gráficas deseadas para representar los parámetros S o el retardo de grupo, entre otros. Existen multitud de opciones para representar los datos, pero puesto que principalmente se desean comparar los parámetros S_{21}, S_{11} y S_{22} y el retardo de grupo negativo, en la Figura B.12 se muestra un tipo de representación apilada, con marcadores para las frecuencias de interés. Por otro lado, en la Figura B.13 se muestran las opciones a configurar para mostrar las gráficas apiladas.



freq1
freq=788.0MHz
Valley

band1
freq=788.0MHz
SP1.SP.S.delay(2,1)=-9.854E-9
dB(SP1.SP.S(2,1))=-27.680
dB(SP1.SP.S(1,1))=-13.237
dB(SP1.SP.S(2,2))=-3.205

bandCenter
freq=1.000GHz
SP1.SP.S.delay(2,1)=8.058E-10
dB(SP1.SP.S(2,1))=-12.291
dB(SP1.SP.S(1,1))=-10.587
dB(SP1.SP.S(2,2))=-10.587

band2
freq=1.212GHz
SP1.SP.S.delay(2,1)=-9.854E-9
dB(SP1.SP.S(2,1))=-27.680
dB(SP1.SP.S(1,1))=-13.237
dB(SP1.SP.S(2,2))=-3.205

freq2
freq=1.212GHz
Valley

Figura B.12: Representación gráfica de las variables

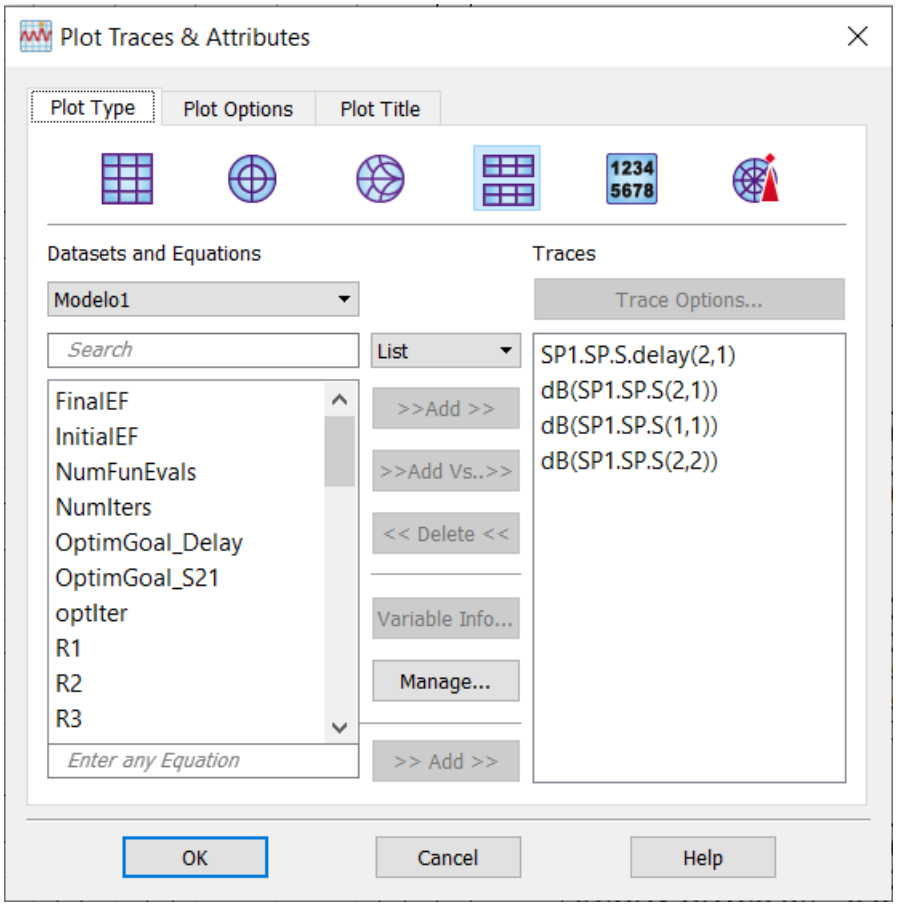


Figura B.13: Opciones para la representación

C. Tutorial Sonnet

Sonnet es una herramienta de diseño y simulación de circuitos electrónicos al igual que ADS. Sin embargo, mientras que en ADS se diseña conectando componentes, en Sonnet se diseña dibujando directamente las dimensiones físicas y teniendo total control sobre las conexiones entre líneas. Además, Sonnet realiza una simulación en onda completa, es decir internamente resuelve las ecuaciones de Maxwell calculando los niveles de voltaje e intensidad en cada punto del dispositivo.

En este Anexo se expone de forma secuencial y mediante las capturas de la interfaz de Sonnet los pasos a seguir para diseñar y simular un circuito.

C.1. Interfaz de Sonnet

Al iniciar Sonnet aparece una pequeña ventana a modo de barra de tareas (Figura C.1) donde se da acceso rápido a las cinco herramientas principales de Sonnet:

- **Edit Proyect:** Creación y modificación de la geometría del circuito.
- **Analyze Proyect:** Lanzamiento, cola, errores y estado de la simulación o simulaciones.
- **View Response:** Visualización de la respuesta del circuito (parámetros S, Y, Z, retardo de grupo...)
- **View Current:** Visualización de las corrientes sobre el diseño geométrico del circuito en 3D.
- **View Far Field:** Visualización del campo lejano. Útil en contextos donde se analiza la directividad de una antena.

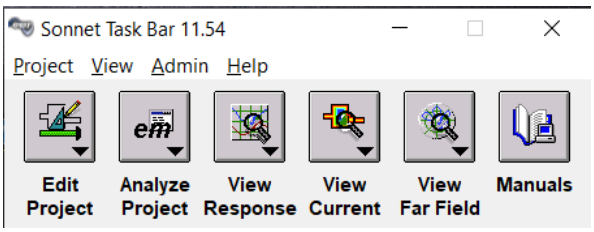


Figura C.1: Ventana principal de Sonnet

C.2. Creación de un prototipo con Sonnet

Para generar un nuevo proyecto se debe seleccionar **New Geometry**, dentro de la opción **Project** de la barra de tareas superior o del botón **Edit Project**. Se generará entonces una ventana como la de la Figura C.2.

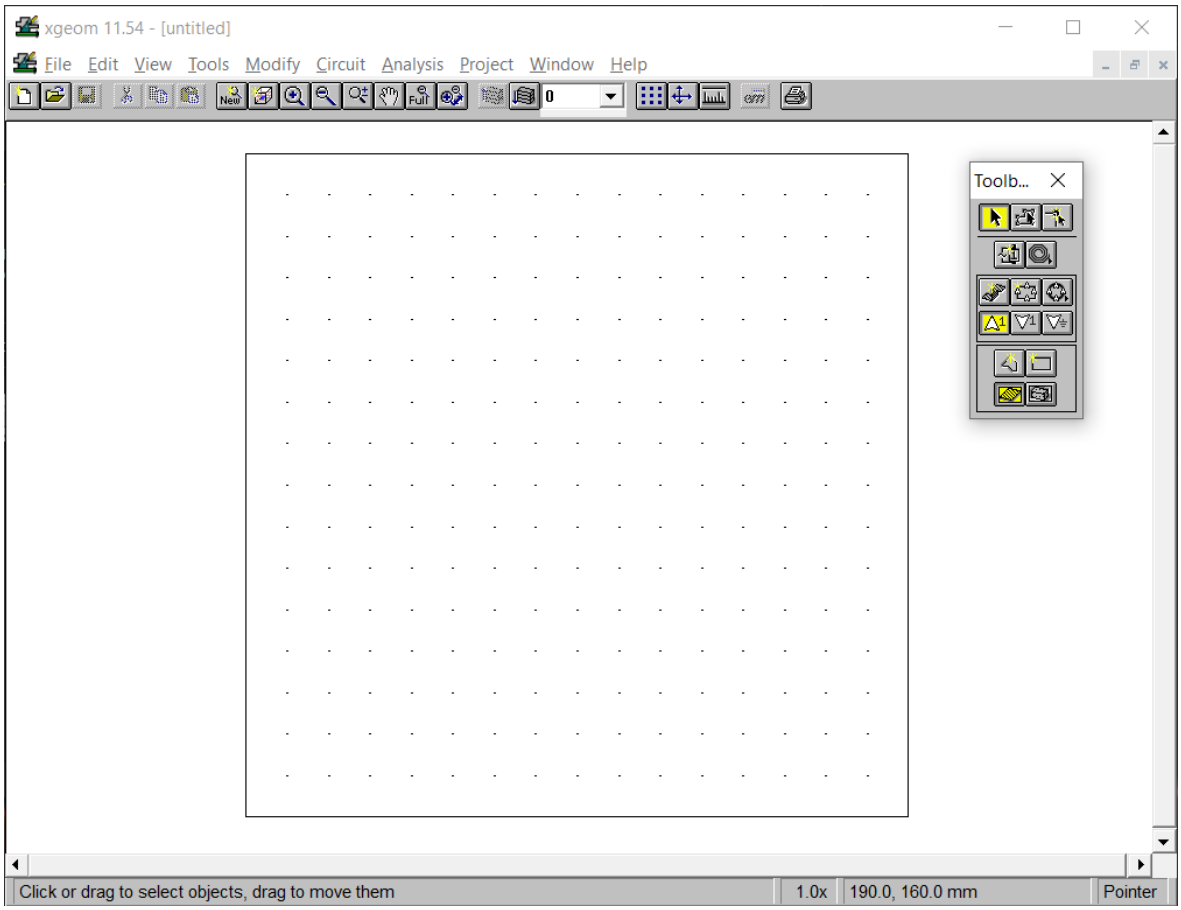


Figura C.2: Editor de Sonnet

C.2.1. Configuración del prototipo

Antes de comenzar a dibujar la geometría de las líneas del prototipo es necesario configurar el proyecto. En el menú **Circuit** de la barra de tareas del Editor (Figura C.2) encontramos diversas opciones que deben configurarse como se explica a continuación.

C.2.1.1. Units

Seleccionar la unidad deseada para cada magnitud. En este TFM se trabaja con la configuración mostrada en la Figura C.3.

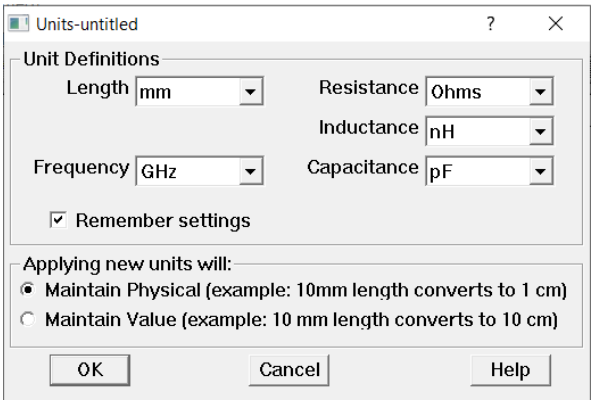


Figura C.3: Configuración de las unidades

C.2.1.2. Box

Configuración de la precisión geométrica indicando el tamaño de celda y los metales superior e inferior de la caja. El metal superior también es posible definirlo como espacio libre si se producen resonancias que interfieran con el prototipo. En este TFM se trabaja con la configuración mostrada en la Figura C.4.

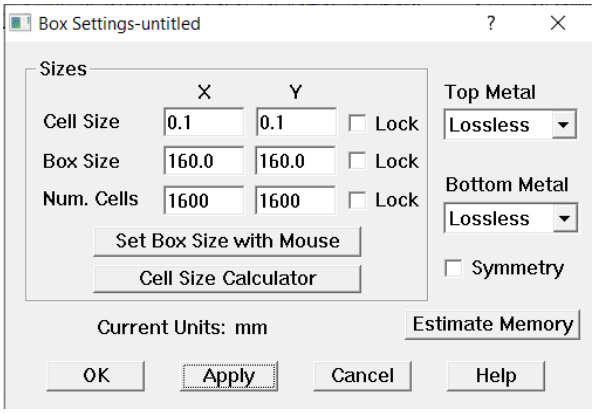
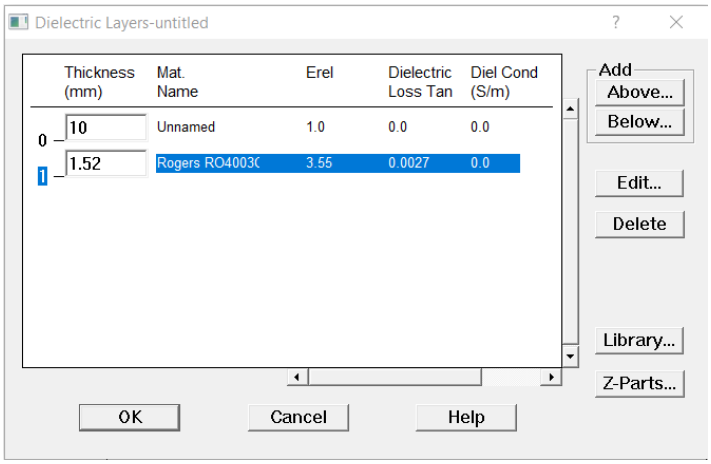


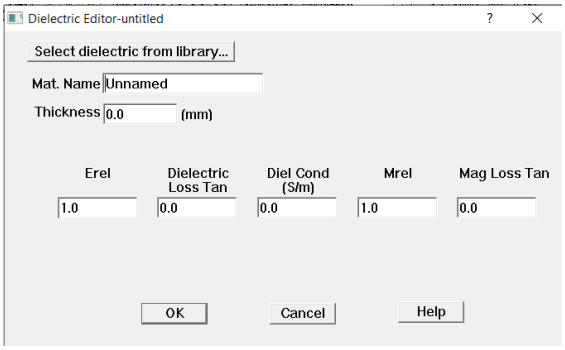
Figura C.4: Configuración de la caja

C.2.1.3. Dielectric Layers

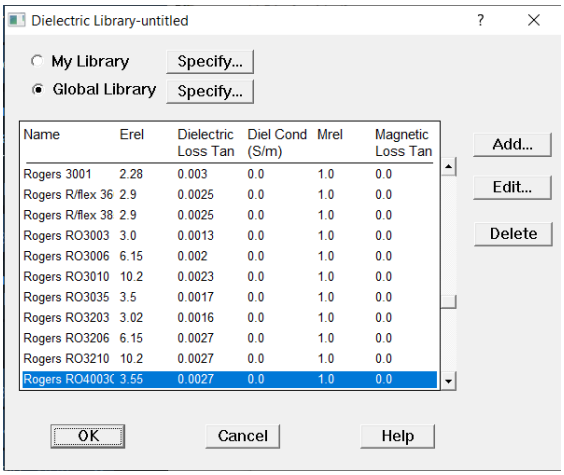
Configuración de la precisión geométrica indicando el tamaño de celda y los metales superior e inferior de la caja. El metal superior también es posible definirlo como espacio libre si se producen resonancias que interfieran con el prototipo. En este TFM se trabaja con la configuración mostrada en la Figura C.5.



(a) Configuración deseada



(b) Editor de capa dieléctrica

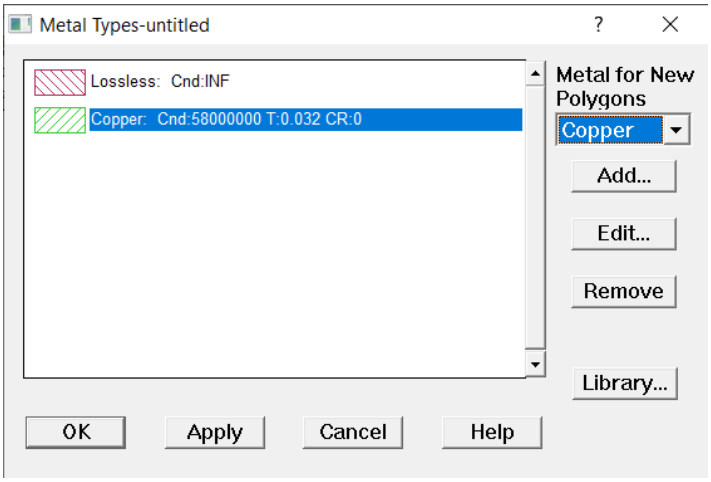


(c) Librería de capas dieléctricas

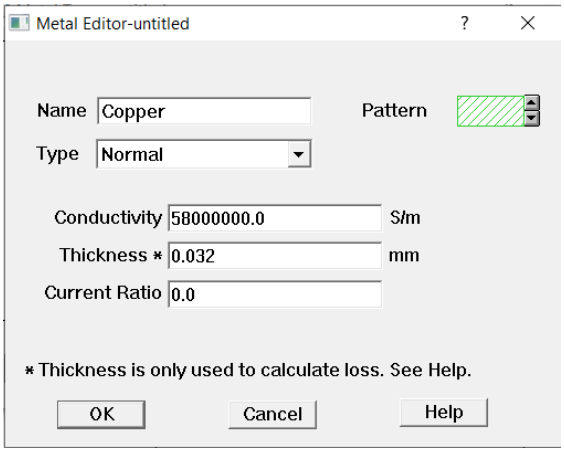
Figura C.5: Configuración de las capas dieléctricas

C.2.1.4. Metal Types

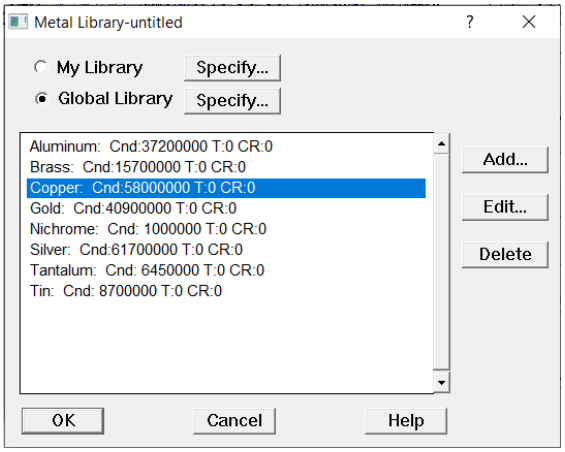
Configuración de los metales conductores del prototipo. En este TFM se trabaja con la configuración mostrada en la Figura C.6.



(a) Configuración deseada



(b) Editor de metales



(c) Librería de metales

Figura C.6: Configuración de los metales conductores

C.2.2. Diseño geométrico del prototipo

C.2.2.1. Añadir una tira microstrip

Para comenzar a diseñar el dispositivo partimos del Editor de Sonnet (Figura C.2). Las tiras microstrip son polígonos geométricos de determinadas dimensiones, por lo que

para generar uno, seleccionamos la opción **Draw Rectangle**, dentro del menú **Add Metalization** que se encuentra en la opción **Tools** de la barra de tareas superior. Con esta opción es posible dibujar el rectángulo, sin embargo también podemos generar un rectángulo a partir de sus dimensiones con la opción **Rectangle** del menú **Add Metalization**. En ambos casos, el resultado será similar al de la Figura C.7. Cabe destacar que el color del polígono añadido representa el tipo de metal que ya se ha configurado con anterioridad en el apartado C.2.1.4: Metal Types.

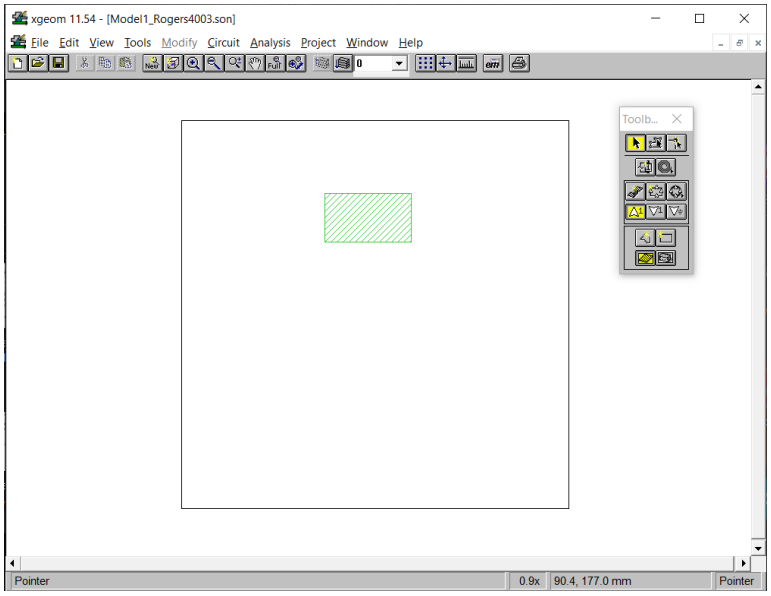


Figura C.7: Línea de transmisión generada mediante un rectángulo

C.2.2.2. Parametrizar las dimensiones de las tiras

Si únicamente se desea diseñar un prototipo no es necesario parametrizar las dimensiones de las tiras ya que no se realizarán muchos cambios más allá de pequeños ajustes para optimizar la respuesta del circuito. Sin embargo, si el objetivo es diseñar diversos prototipos, merece la pena parametrizar las dimensiones para poder variar todo el diseño únicamente introduciendo las nuevas medidas.

La versión de Sonnet utilizada no implementa la opción de definir dimensiones en base a otras mediante una fórmula, pero sí que permite nombrar dimensiones de diversos polígonos con el mismo nombre para que varíen de igual forma y definir puntos de anclaje que serán fijos aunque varíe dicha dimensión.

Definir todo un circuito parametrizándolo geométricamente puede ser costoso con esta versión de Sonnet ya que no es demasiado intuitiva. A continuación se explica de forma secuencial:

1. Seleccionamos **Tools →Add Parameter →Add Anchored**.

2. Seleccionamos 3 nodos (Figura C.8) teniendo en cuenta que:
   - El primer nodo (cuadrado hueco) será el punto de anclaje.
   - El segundo nodo marca la dimensión a parametrizar.
   - El tercer nodo es un nodo relativo (cuadrado sólido), sirve para indicar que también variará su posición junto con el segundo.
3. Pulsamos Enter y aparecerá un diálogo donde indicamos el nombre de dicho parámetro y, si se desea, variar la dimensión (Figura C.9).
4. Es posible modificar los nodos seleccionando **Modify Point Set** del menú contextual desplegado al pulsar sobre el parámetro creado con el botón derecho.
5. Para añadir nuevos nodos relativos (como el tercero) debemos mantener pulsada la tecla Control mientras seleccionamos estos nuevos nodos.

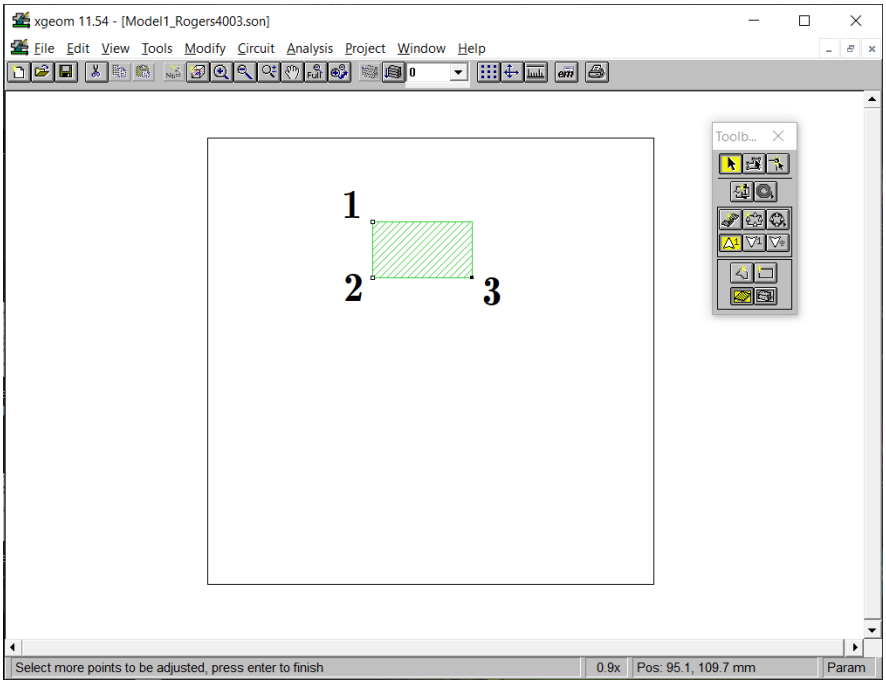


Figura C.8: Selección de nodos a parametrizar mediante el método de anclado

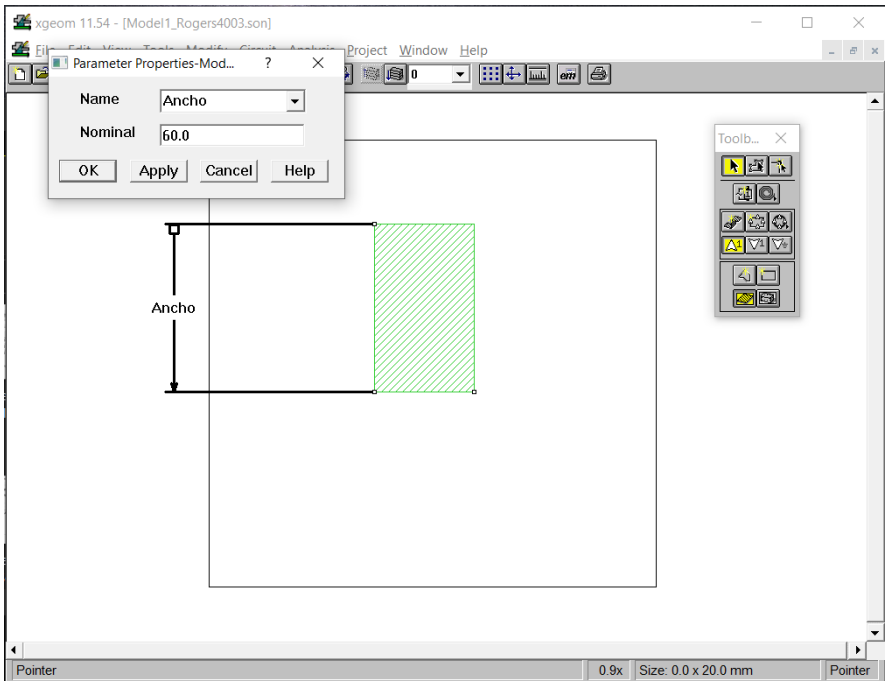


Figura C.9: Diálogo de creación el parámetro

También es interesante la parametrización simétrica, ya que no ancla un nodo a un punto fijo, sino que ancla el punto medio del polígono. La creación de dichos parámetros es algo diferente a la anterior:

1. Seleccionamos **Tools →Add Parameter →Add Symetric**.
2. Seleccionamos 4 nodos (Figura C.10) teniendo en cuenta que:
   - El primer y segundo nodo serán relativos entre sí y se desplazarán por igual.
   - Tras seleccionarlos debemos pulsar Enter para indicar que cambiamos de posición.
   - El tercer y cuarto nodo también serán relativos entre sí.
   - Cada grupo de nodos se desplazará de forma simétrica respecto al centro de la figura.
3. Pulsamos enter y aparecerá de nuevo el diálogo de propiedades(Figura C.11).
4. Es posible modificar los grupos de nodos seleccionando **Modify Left Point Set** o **Modify Right Point Set** del menú contextual.
5. Para añadir nuevos nodos relativos al grupo debemos mantener pulsada la tecla Control mientras seleccionamos estos nuevos nodos.

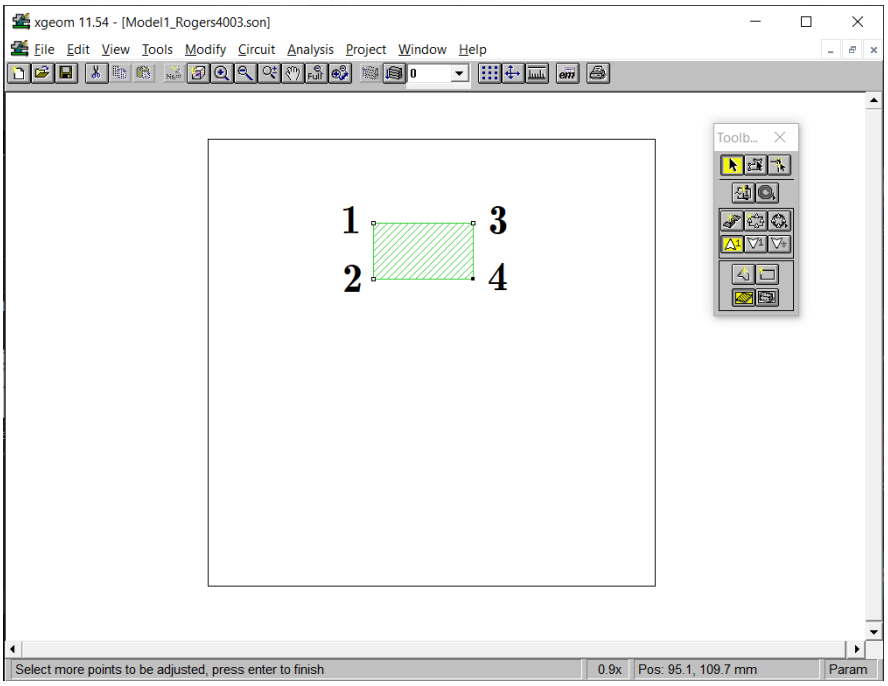


Figura C.10: Selección de nodos a parametrizar mediante el método simétrico

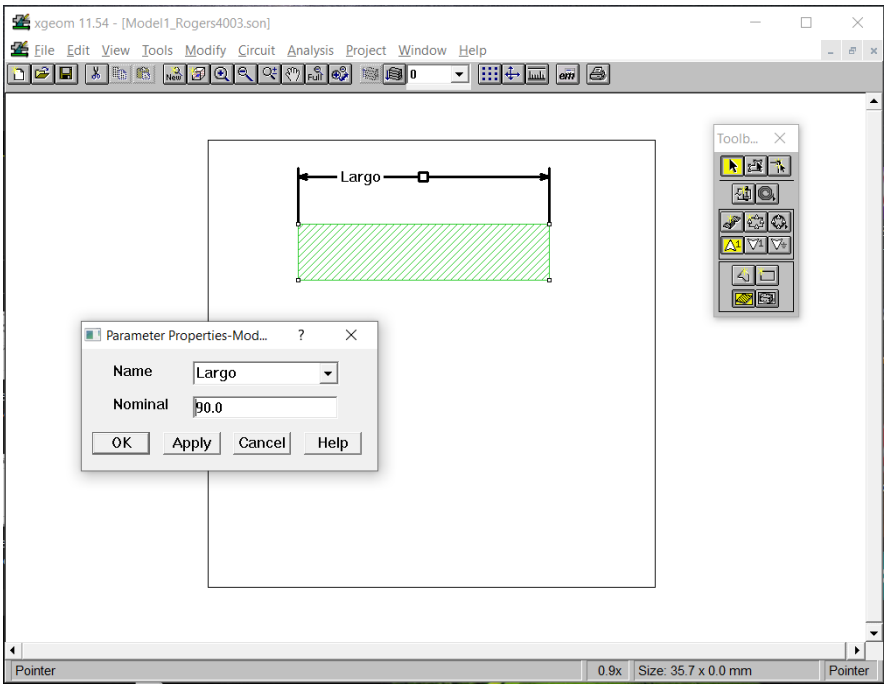


Figura C.11: Diálogo de creación el parámetro

C.2.2.3. Añadir los Puertos

Para poder simular el dispositivo es necesario indicar dónde se encuentran los puertos (únicamente dos en este TFM). Para ello es necesario crear tiras microstrip que se encuentren en contacto con el borde de la caja (Figura C.12). Esas tiras deben tener una anchura equivalente a 50Ω o a la impedancia característica deseada para los puertos.

Tras generar el polígono y asegurar que está en contacto con el borde, se procede a añadir los puertos mediante la opción **Add Ports** del menú **Tools** de la barra de herramientas del Editor. Esto generará un indicador numérico en dicha posición que simboliza el puerto. Si se hace doble click sobre él se accede al menú de configuración (Figura C.13).

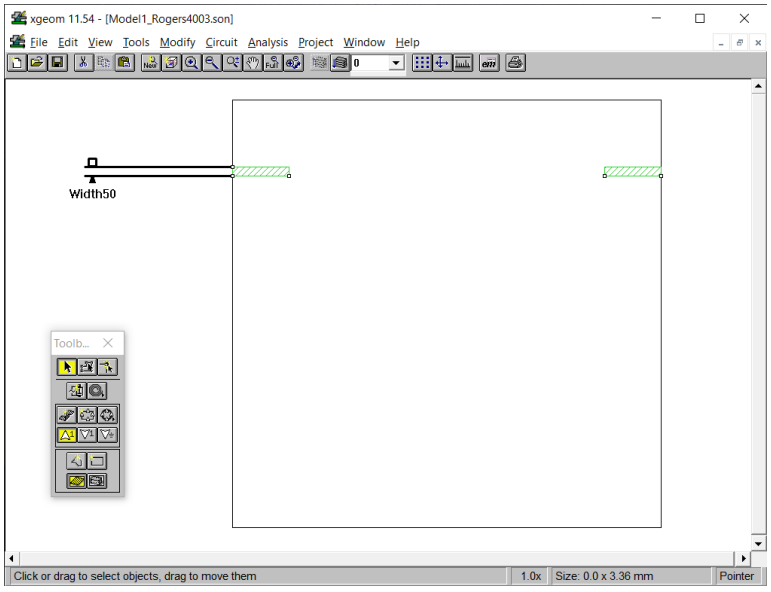


Figura C.12: Creación de tiras para situar los puertos

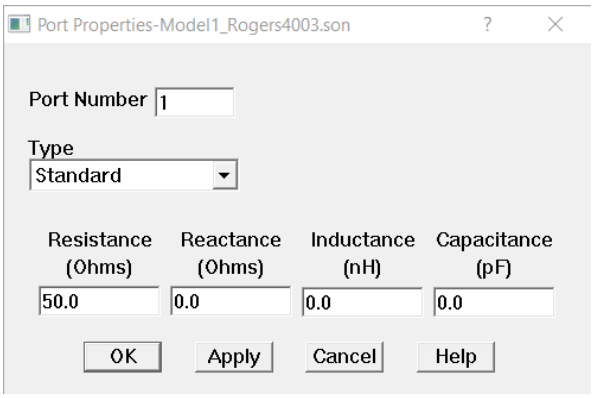


Figura C.13: Diálogo de configuración del puerto

C.2.2.4. Añadir Resistencias

Para finalizar este apartado se explica cómo añadir resistencias. Para añadir un componente nuevo se selecciona la opción **Ideal** del menú **Add Component**, que se encuentra en el apartado **Tools** de la barra de herramientas. Hay que tener en cuenta el tipo y modelo de resistencias que se va a utilizar para determinar el ancho del terminal ($0,5$mm en este TFM). También es posible decidir si se tratará al componente de forma ideal o no, que al tratarse de resistencias de valores acotados en este caso es posible analizar con una resistencia ideal. En la Figura C.14 se muestra el diálogo de configuración.

Es importante tener en cuenta el lugar en el que se añade la resistencia respecto a las tiras, siendo aconsejable centrarlas respecto a sus bordes.

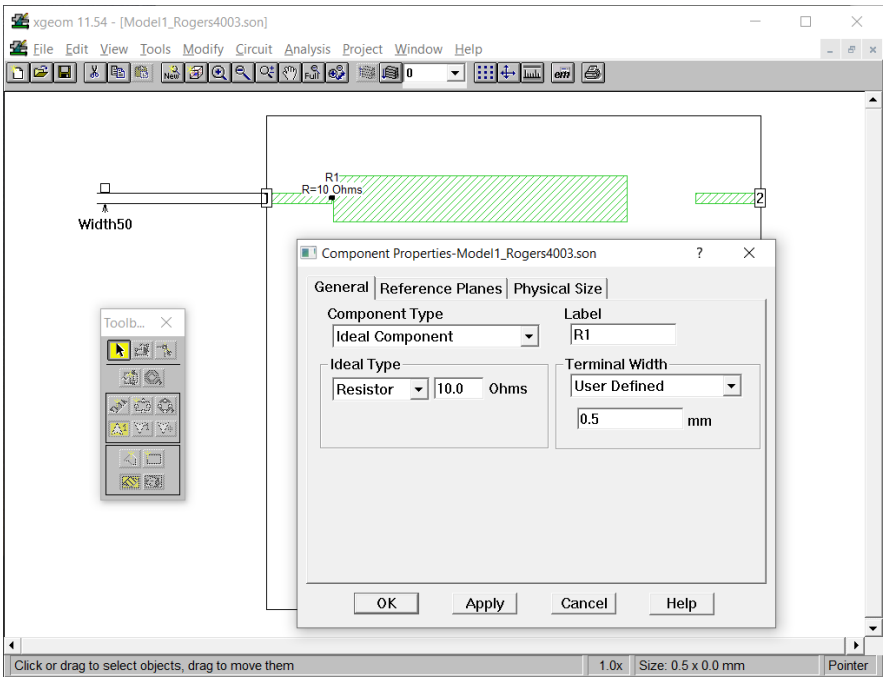


Figura C.14: Añadir un componente al prototipo

C.2.2.5. Añadir Planos de Referencia

Los planos de referencia permiten que el software calcule el retardo de grupo con respecto a dichos puntos, descontando el retardo debido a la longitud de los puertos. Para añadir los planos de referencia seleccionamos la opción **Ref. Planes/Calc Length...** del menú **Circuit**. Existen tres opciones para marcar el plano de referencia, pero la más sencilla es **Linked**, que nos permite simplemente pinchar en el punto donde queremos que se sitúe dicho plano y el software calcula el resto (Figura C.15).

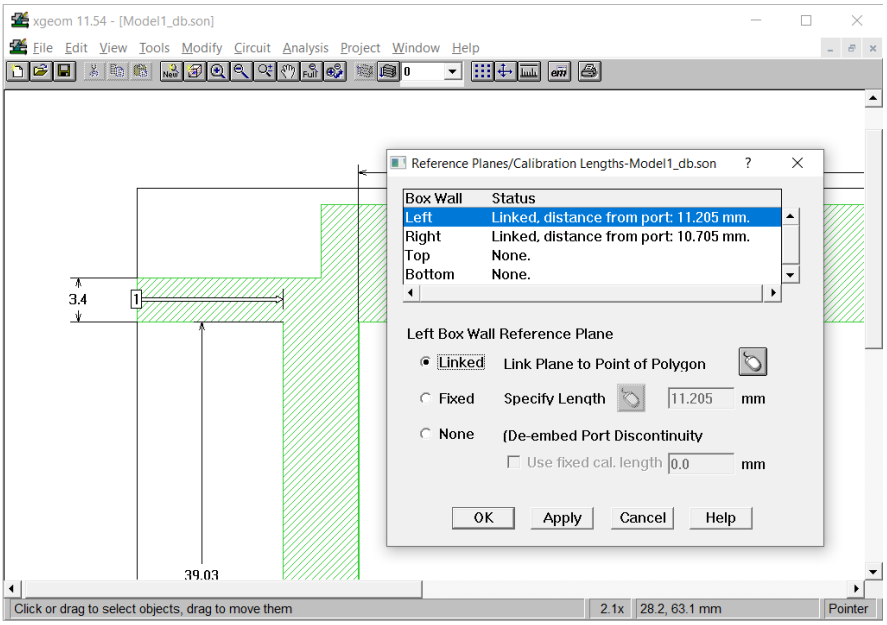


Figura C.15: Añadir un plano de referencia

C.2.3. Configuración de la simulación

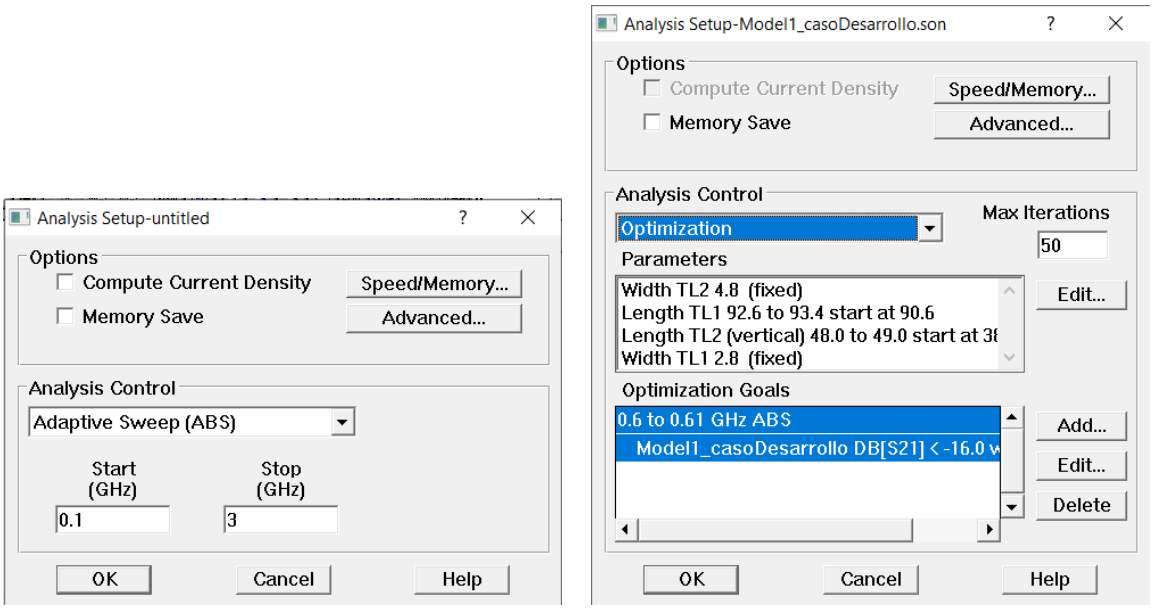
Para poder simular correctamente, es necesario configurar los parámetros de análisis seleccionando, al menos, el tipo y el rango frecuencial. Esta configuración se encuentra en el En este TFM se opta por **Adaptative Sweep (ABS)** (Figura C.16a). Este tipo de análisis simula el dispositivo para las frecuencias inicial y final introducidas por el usuario. Posteriormente y de forma iterativa, calcula la respuesta del circuito en otros puntos hasta que determina un polinomio racional para los parámetros S.

También es posible elegir un tipo de análisis para optimizar el circuito en base a unas condiciones (Figura C.16b).

Es interesante también las opciones de cálculo de densidad de corriente y de ahorro de memoria. La primera, como su nombre indica, devuelve también la densidad de corriente para el dispositivo, pudiendo visualizarla mediante una vista 3D del prototipo. La segunda, al igual que las opciones dentro del menú **Speed/Memory**, sirven para realizar simulaciones más precisas a costa de un uso intensivo de la memoria o ahorrar memoria a costa de sacrificar la precisión.

C.2.4. Simulación y Visualización

Tras analizar el prototipo (**Project →Analyze**) se obtendrá un resultado similar al de la Figura C.17, es posible visualizar la respuesta mediante la opción **View Response**, de la barra de iconos superior, se abre una nueva ventana con la respuesta (Figura C.17). Para añadir otros parámetros S (o aquellos que se deseen), es necesario



(a) Configuración análisis ABS **(b)** Configuración análisis Optimización

Figura C.16: Configuración del análisis

abrir el menú **Add Curve Group** de la opción **Curve** de la barra de herramientas. Sin embargo, si lo que se desea es representar el retardo de grupo, que se calcula a partir del parámetro S_{21}, se debe añadir mediante el diálogo que se muestra al seleccionar **Add Equation Curve** del menú **Equation**. Tras añadir todas las respuestas deseadas se obtiene una representación similar a la de la Figura C.18

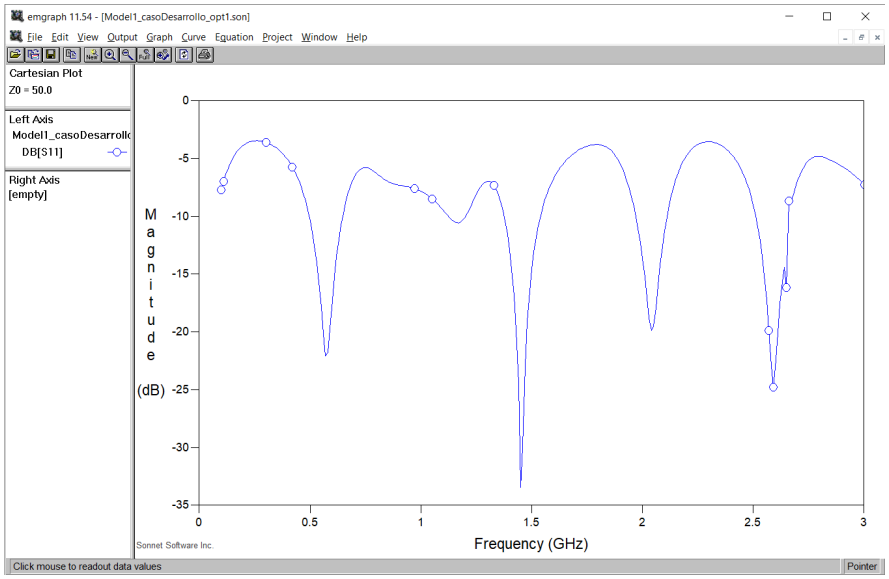


Figura C.17: Entorno de visualización

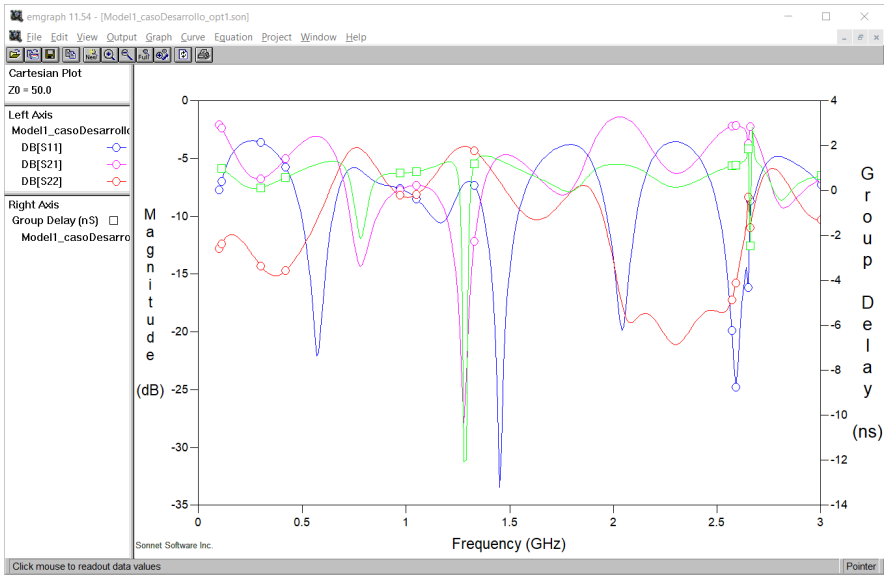


Figura C.18: Entorno de visualización con múltiples respuestas

D. Códigos Matlab

En este anexo se detallan los códigos desarrollados en Matlab para este TFM. Cabe destacar que las llamadas a la función **setfigpaper** ejecutan una función almacenada en local y de licencia GPL 3.0 [46].

Código D.1: Script de simulación del Modelo 1

```
%% Test Model 1
addpath('C:\Users\Marga\Documents\TFM\Computacion\MATLAB\funcs');

%% Data
allvars = {5,55,5,5,40,100};
r1 = allvars{1};
z1 = allvars{2};
r2 = allvars{3};
r3 = allvars{4};
z2 = allvars{5};
r4 = allvars{6};

z0=50;
f0=1e9; % 1 GHz
fn = 0:0.001:2; % 1 MHz step for 1 GHz
f=(fn.*f0).';

%% Calc
s = fcalcMod1(r1,z1,r2,r3,z2,r4,z0,fn);
s21=reshape(s(2,1,:),[],1,1);
s11=reshape(s(1,1,:),[],1,1);
s22=reshape(s(2,2,:),[],1,1);
group_delay = -(diff(unwrap(angle(s21),pi))./diff(f))/(2*pi);
fig = figure;
left_color = [0 0 0];
right_color = [0 0 0];
set(fig,'defaultAxesColorOrder',[left_color; right_color]);
title("Modelo 1: R1="+num2str(allvars{1})+" Z1="+num2str(allvars{2})+...
    " R2="+num2str(allvars{3})+" R3="+num2str(allvars{4})+...
    " Z2="+num2str(allvars{5})+" R4="+num2str(allvars{6}));
hold on
grid on
yyaxis left
plot(f(1:end-1).*1e-9,group_delay.*1e9,'b-');
ylabel('NGD (ns)')
yyaxis right
plot(f(1:end-1).*1e-9,20*log10(abs(s21(1:end-1))),'r-');
```

```
plot(f(1:end-1).*1e-9,20*log10(abs(s11(1:end-1))),'m-');
plot(f(1:end-1).*1e-9,20*log10(abs(s22(1:end-1))),'k-');
ylabel('SP (dB)');
xlabel('f (GHz)');
legend('NGD','S21','S11','S22','Location','best');
yyaxis right
ylim([-35, 5])
yyaxis left
ylim([-35, 5])
setfigpaper;
```

Código D.2: Función para el cálculo de los parámetros S del Modelo 1

```
function [S] = fcalcMod1(r1,z1,r2,r3,z2,r4,z0,fn)
%% Initial variables
if (z1~=0); y1=1/z1; else; y1=0; end
if (z2~=0); y2=1/z2; else; y2=0; end

theta1a=pi;
theta2a=2*pi;
Y0=1/z0;

%% Create theta matrix
THETA1A = reshape(theta1a.*fn,1,1,[]);
THETA2A = reshape(theta2a.*fn,1,1,[]);

%% ABCD matrix for Resistances and Transmission Lines
R1 = [1 r1; 0 1];
R2 = [1 r2; 0 1];
R3 = [1 r3; 0 1];
R4 = [1 r4; 0 1];
% Adecuation for matrix operations
R1=repmat(R1,1,1,length(fn));
R2=repmat(R2,1,1,length(fn));
R3=repmat(R3,1,1,length(fn));
R4=repmat(R4,1,1,length(fn));
% ABCD matrix for TLs
TL1 = [cos(THETA1A) 1i*z1*sin(THETA1A); 1i*y1*sin(THETA1A) cos(THETA1A)];
TL2 = [cos(THETA2A) 1i*z2*sin(THETA2A); 1i*y2*sin(THETA2A) cos(THETA2A)];

%% Calc Paths equivalent
PATH1 = zeros(size(TL1));
PATH2 = zeros(size(TL1));
% tic
for m=1:length(TL1)
        PATH1(:,:,m) = R1(:,:,m)*TL1(:,:,m)*R2(:,:,m);
        PATH2(:,:,m) = R3(:,:,m)*TL2(:,:,m)*R4(:,:,m);
end
S = fPaths2SP(PATH1, PATH2, Y0);
end
```

Código D.3: Función para el cálculo de la matriz SP equivalente de una red de dos ramas en paralelo

```
function [S] = fPaths2SP(PATH1, PATH2, Y0)
% Conversion ABCD -> Y matrix
A1 = PATH1(1,1,:); B1 = PATH1(1,2,:); C1 = PATH1(2,1,:); D1 = PATH1(2,2,:);
A2 = PATH2(1,1,:); B2 = PATH2(1,2,:); C2 = PATH2(2,1,:); D2 = PATH2(2,2,:);
Y_path1 = [(D1./B1),((B1.*C1-A1.*D1)./(B1));(-1/B1),(A1./B1)];
Y_path2 = [(D2./B2),((B2.*C2-A2.*D2)./(B2));(-1./B2),(A2./B2)];
% Y_path1 = abcd2y(PATH1);
% Y_path2 = abcd2y(PATH2);

% Calc circuit equivalent Y matrix
Y = Y_path1 + Y_path2;

% Conversoin Y -> S matrix
Y11 = Y(1,1,:); Y12 = Y(1,2,:); Y21 = Y(2,1,:); Y22 = Y(2,2,:);
deltaY = (Y11+Y0).*(Y22+Y0)-Y12.*Y21;
S11(1,1,:) = ((Y0-Y11).*(Y0+Y22)+Y12.*Y21)./(deltaY);
S12(1,1,:) = (-2*Y12.*Y0)./(deltaY);
S21(1,1,:) = (-2*Y21.*Y0)./(deltaY);
S22(1,1,:) = ((Y0+Y11).*(Y0-Y22)+Y12.*Y21)./(deltaY);
S = [S11, S12; S21, S22];
% S = y2s(Y,z0);
end
```

Código D.4: Creación del DataStore

```
%% MacroDS
% Convertir un conjunto de archivos .mat independientes en un datastore
% para posteriormente almacenarlo como un tallDataStore y poder modificarlo
inDir = "E:\TFM_DATA\Mod1";
varnames = load('varnamesModels.mat');
varnames = varnames.varnamesMod1;
outDir = "E:\TFM_DATA\Mod1DS";
% Read the data back in as a tall array. First create a datastore ...
ds = fileDatastore(fullfile(inDir, '*.mat'), ...
    'ReadFcn', @(fname) [repmat(str2double(fname(24:26)),161051,1), ...
    repmat(str2double(fname(29:31)),161051,1) ,...
    repmat(str2double(fname(34:36)),161051,1) ,...
    getfield(load(fname), 'data')], ...
    'UniformRead', true);
% ... and then a tall array
ds = transform(ds, @(x) array2table(x,'VariableNames',varnames));
tdata = tall(ds);
write(outDir, tdata);
```

Código D.5: Filtrado del DataStore Modelo 1

```
%% Create filtered DataStores
% Add functions path
addpath('funcs');
```

```
% Load Datastore
ds = datastore('E:\TFM_DATA\Mod1DS\');

%% Double Band
cDoubleBand = @(x) (x.Del1<=-1e-9 & x.S211>=0.1 & isnan(x.DelC));
ds.reset;
tic
[newData] = parforApplyCondition(ds,cDoubleBand);
toc
tdata = tall(newData);
outDir = "E:\TFM_DATA\Mod1DS_DoubleBand";
write(outDir, tdata);

%% Triple Band
cTripleBand = @(x) (x.Del1<=-1e-9 & x.S211>=0.1 & x.DelC<=-1e-9 & x.S21C>=0.1);
ds.reset;
tic
[newData] = parforApplyCondition(ds,cTripleBand);
toc
tdata = tall(newData);
outDir = "E:\TFM_DATA\Mod1DS_TripleBand";
write(outDir, tdata);

%% Center Band
cCenterBand = @(x) (x.DelC<=-1e-9 & x.S21C>=0.1 & isnan(x.Del1));
ds.reset;
tic
[newData] = parforApplyCondition(ds,cCenterBand);
toc
tdata = tall(newData);
outDir = "E:\TFM_DATA\Mod1DS_CenterBand";
write(outDir, tdata);
```

Código D.6: Aplicación para el análisis paramétrico del Modelo 1

```
classdef visualizerMod1 < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure matlab.ui.Figure
        GridLayout matlab.ui.container.GridLayout
        LeftPanel matlab.ui.container.Panel
        OpenButton matlab.ui.control.Button
        ApplyConditionButton matlab.ui.control.Button
        FolderPathEditFieldLabel matlab.ui.control.Label
        FolderPathEditField matlab.ui.control.EditField
        DataConditionDropDownLabel matlab.ui.control.Label
        DataConditionDropDown matlab.ui.control.DropDown
        PersonalizedDataConditionTextAreaLabel matlab.ui.control.Label
        PersonalizedDataConditionTextArea matlab.ui.control.TextArea
        TitleEditFieldLabel matlab.ui.control.Label
```

```
17          TitleEditField matlab.ui.control.EditField
18          RightPanel matlab.ui.container.Panel
19          VisualizeButton matlab.ui.control.Button
20          XAxisDropDownLabel matlab.ui.control.Label
21          XAxisDropDown matlab.ui.control.DropDown
22          PersonalizedXAxisTextAreaLabel matlab.ui.control.Label
23          PersonalizedXAxisTextArea matlab.ui.control.TextArea
24          YAxisDropDownLabel matlab.ui.control.Label
25          YAxisDropDown matlab.ui.control.DropDown
26          PersonalizedYAxisTextAreaLabel matlab.ui.control.Label
27          PersonalizedYAxisTextArea matlab.ui.control.TextArea
28          ZAxisDropDownLabel matlab.ui.control.Label
29          ZAxisDropDown matlab.ui.control.DropDown
30          PersonalizedZAxisTextAreaLabel matlab.ui.control.Label
31          PersonalizedZAxisTextArea matlab.ui.control.TextArea
32      end
33
34
35      properties (Access = private)
36          % S$_{21}$($f_{1}$) [dB]
37          % ngd($f_{1}$) [s]
38
39
40          onePanelWidth = 576;
41
42          cDoubleBand = @(x) (x.Del1<=-1e-9 & x.S211>=0.1);
43          cCenterBand = @(x) (x.DelC<=-1e-9 & x.S21C>=0.1);
44          cBothOR = @(x) (x.Del1<=-1e-9 & x.S211>=0.1) | (x.DelC<=-1e-9 & x.S21C↩
                ↪ >=0.1);
45          cBothAND = @(x) (x.Del1<=-1e-9 & x.S211>=0.1) & (x.DelC<=-1e-9 & x.S21C↩
                ↪ >=0.1);
46          cNone = @(x) 1:height(x);
47          c4Rs = @(x) (x.R1~=0 & x.R2~=0 & x.R3~=0 & x.R4~=0);
48
49          condX = @(x) x.iDel1;
50          condY = @(x) x.iDel1;
51          condZ = @(x) x.iDel1;
52          condData = @(x) (x.Del1<=-1e-9 & x.S211>=0.1);
53
54          ds
55          tData
56
57          plt3d = 0
58      end
59
60      methods (Access = private)
61
62          % Changes arrangement of the app based on UIFigure width
63          function updateAppLayout(app, event)
64              currentFigureWidth = app.UIFigure.Position(3);
65              if(currentFigureWidth <= app.onePanelWidth)
```

```
66                  % Change to a 2x1 grid
67                  app.GridLayout.RowHeight = {480, 480};
68                  app.GridLayout.ColumnWidth = {'1x'};
69                  app.RightPanel.Layout.Row = 2;
70                  app.RightPanel.Layout.Column = 1;
71              else
72                  % Change to a 1x2 grid
73                  app.GridLayout.RowHeight = {'1x'};
74                  app.GridLayout.ColumnWidth = {'1x', '1x'};
75                  app.RightPanel.Layout.Row = 1;
76                  app.RightPanel.Layout.Column = 2;
77              end
78          end
79
80          function [newData] = parforApplyCondition(app, ds,cond)
81              N = numpartitions(ds,gcp);
82              newData = [];
83              parfor ii = 1:N
84                  subds = partition(ds,N,ii);
85                  Data = [];
86                  while hasdata(subds)
87                      data = read(subds);
88                      data = data(cond(data),:);
89                      Data = [Data; data];
90                  end
91                  newData = [newData; Data];
92              end
93          end
94      end
95
96
97      % Callbacks that handle component events
98      methods (Access = private)
99
100         % Code that executes after component creation
101         function startupFcn(app)
102             app.UIFigure.SizeChangedFcn = createCallbackFcn(app, @updateAppLayout, ↩
                    ↪ true);
103             app.XAxisDropDown.Items = {'iDel1', 'iDelC', 'Del1' 'DelC' 'S211' 'S21C' 'BW1'↩
                    ↪ , 'BWC' 'BW1dB1' 'BW1dBC' 'BW3dB1' 'BW3dBC' 'S111' 'S11C' 'S221↩
                    ↪ ' 'S22C' 'f2/f1' 'Personalized'};
104             app.YAxisDropDown.Items = {'iDel1', 'iDelC', 'Del1' 'DelC' 'S211' 'S21C' 'BW1'↩
                    ↪ , 'BWC' 'BW1dB1' 'BW1dBC' 'BW3dB1' 'BW3dBC' 'S111' 'S11C' 'S221↩
                    ↪ ' 'S22C' 'f2/f1' 'Personalized'};
105             app.ZAxisDropDown.Items = {'None' 'iDel1', 'iDelC', 'Del1' 'DelC' 'S211' 'S21C'↩
                    ↪  'BW1', 'BWC' 'BW1dB1' 'BW1dBC' 'BW3dB1' 'BW3dBC' 'S111' '↩
                    ↪ S11C' 'S221' 'S22C' 'f2/f1' 'Personalized'};
106             app.DataConditionDropDown.Items = {'Double Band', 'Center Band', 'Both (↩
                    ↪ OR)', 'Both (AND)', '4 Resistances', 'None', 'Personalized'};
107             % app.ds = datastore('E:\TFM_DATA\Mod1DS\');
108             % app.ds.reset;
```

```
109             % addpath('C:\Users\Marga\Documents\TFM\Analisis\MATLAB\funcs');
110             app.PersonalizedXAxisTextArea.Value = func2str(app.condX);
111             app.PersonalizedYAxisTextArea.Value = func2str(app.condY);
112             app.PersonalizedDataConditionTextArea.Value = func2str(app.condData);
113         end
114
115         % Value changed function: XAxisDropDown
116         function XAxisDropDownValueChanged(app, event)
117             value = app.XAxisDropDown.Value;
118             app.PersonalizedXAxisTextArea.Editable = ”off”;
119             switch value
120                 case 'iDel1'
121                     app.condX = @(x) x.iDel1;
122                 case 'iDelC'
123                     app.condX = @(x) x.iDelC;
124                 case 'Del1'
125                     app.condX = @(x) x.Del1;
126                 case 'DelC'
127                     app.condX = @(x) x.DelC;
128                 case 'S211'
129                     app.condX = @(x) 20*log10(x.S211);
130                 case 'S21C'
131                     app.condX = @(x) 20*log10(x.S21C);
132                 case 'S221'
133                     app.condX = @(x) 20*log10(x.S221);
134                 case 'S22C'
135                     app.condX = @(x) 20*log10(x.S22C);
136                 case 'S111'
137                     app.condX = @(x) 20*log10(x.S111);
138                 case 'S11C'
139                     app.condX = @(x) 20*log10(x.S11C);
140                 case 'BW1'
141                     app.condX = @(x) x.iBW12 - x.iBW11;
142                 case 'BWC'
143                     app.condX = @(x) x.iBWC2 - x.iBWC1;
144                 case 'BW1dB1'
145                     app.condX = @(x) x.iBW1dB12 - x.iBW1dB11;
146                 case 'BW1dBC'
147                     app.condX = @(x) x.iBW1dBC2 - x.iBW1dBC1;
148                 case 'BW3dB1'
149                     app.condX = @(x) x.iBW3dB12 - x.iBW3dB11;
150                 case 'BW3dBC'
151                     app.condX = @(x) x.iBW3dBC2 - x.iBW3dBC1;
152                 case 'f2/f1'
153                     app.condX = @(x) (2000-x.iDel1)./x.iDel1;
154                 case 'Personalized'
155                     app.PersonalizedXAxisTextArea.Editable = ”on”;
156                     app.PersonalizedXAxisTextAreaValueChanged;
157             end
158             app.PersonalizedXAxisTextArea.Value = func2str(app.condX);
159         end
```

```
160
161         % Value changed function: YAxisDropDown
162         function YAxisDropDownValueChanged(app, event)
163             value = app.YAxisDropDown.Value;
164             app.PersonalizedYAxisTextArea.Editable = ”off”;
165             switch value
166                 case 'iDel1'
167                     app.condY = @(x) x.iDel1;
168                 case 'iDelC'
169                     app.condY = @(x) x.iDelC;
170                 case 'Del1'
171                     app.condY = @(x) x.Del1;
172                 case 'DelC'
173                     app.condY = @(x) x.DelC;
174                 case 'S211'
175                     app.condY = @(x) 20*log10(x.S211);
176                 case 'S21C'
177                     app.condY = @(x) 20*log10(x.S21C);
178                 case 'S221'
179                     app.condY = @(x) 20*log10(x.S221);
180                 case 'S22C'
181                     app.condY = @(x) 20*log10(x.S22C);
182                 case 'S111'
183                     app.condY = @(x) 20*log10(x.S111);
184                 case 'S11C'
185                     app.condY = @(x) 20*log10(x.S11C);
186                 case 'BW1'
187                     app.condY = @(x) x.iBW12 - x.iBW11;
188                 case 'BWC'
189                     app.condY = @(x) x.iBWC2 - x.iBWC1;
190                 case 'BW1dB1'
191                     app.condY = @(x) x.iBW1dB12 - x.iBW1dB11;
192                 case 'BW1dBC'
193                     app.condY = @(x) x.iBW1dBC2 - x.iBW1dBC1;
194                 case 'BW3dB1'
195                     app.condY = @(x) x.iBW3dB12 - x.iBW3dB11;
196                 case 'BW3dBC'
197                     app.condY = @(x) x.iBW3dBC2 - x.iBW3dBC1;
198                 case 'f2/f1'
199                     app.condY = @(x) (2000-x.iDel1)./x.iDel1;
200                 case 'Personalized'
201                     app.PersonalizedYAxisTextArea.Editable = ”on”;
202                     app.PersonalizedYAxisTextAreaValueChanged;
203             end
204             app.PersonalizedYAxisTextArea.Value = func2str(app.condY);
205         end
206
207         % Value changed function: ZAxisDropDown
208         function ZAxisDropDownValueChanged(app, event)
209             value = app.ZAxisDropDown.Value;
210             app.PersonalizedZAxisTextArea.Editable = ”off”;
```

```
211             app.plt3d = 1;
212             switch value
213                 case 'None'
214                     app.plt3d = 0;
215                     app.condZ = @(x) x.iDel1;
216                 case 'iDel1'
217                     app.condZ = @(x) x.iDel1;
218                 case 'iDelC'
219                     app.condZ = @(x) x.iDelC;
220                 case 'Del1'
221                     app.condZ = @(x) x.Del1;
222                 case 'DelC'
223                     app.condZ = @(x) x.DelC;
224                 case 'S211'
225                     app.condZ = @(x) 20*log10(x.S211);
226                 case 'S21C'
227                     app.condZ = @(x) 20*log10(x.S21C);
228                 case 'S221'
229                     app.condZ = @(x) 20*log10(x.S221);
230                 case 'S22C'
231                     app.condZ = @(x) 20*log10(x.S22C);
232                 case 'S111'
233                     app.condZ = @(x) 20*log10(x.S111);
234                 case 'S11C'
235                     app.condZ = @(x) 20*log10(x.S11C);
236                 case 'BW1'
237                     app.condZ = @(x) x.iBW12 - x.iBW11;
238                 case 'BWC'
239                     app.condZ = @(x) x.iBWC2 - x.iBWC1;
240                 case 'BW1dB1'
241                     app.condZ = @(x) x.iBW1dB12 - x.iBW1dB11;
242                 case 'BW1dBC'
243                     app.condZ = @(x) x.iBW1dBC2 - x.iBW1dBC1;
244                 case 'BW3dB1'
245                     app.condZ = @(x) x.iBW3dB12 - x.iBW3dB11;
246                 case 'BW3dBC'
247                     app.condZ = @(x) x.iBW3dBC2 - x.iBW3dBC1;
248                 case 'f2/f1'
249                     app.condZ = @(x) (2000-x.iDel1)./x.iDel1;
250                 case 'Personalized'
251                     app.PersonalizedZAxisTextArea.Editable = ”on”;
252                     app.PersonalizedZAxisTextAreaValueChanged;
253             end
254             app.PersonalizedZAxisTextArea.Value = func2str(app.condZ);
255         end
256 
257         % Value changed function: PersonalizedXAxisTextArea
258         function PersonalizedXAxisTextAreaValueChanged(app, event)
259             value = app.PersonalizedXAxisTextArea.Value;
260             try
261                 app.condX = str2func(value{:});
```

```
262             catch ME
263                 str = sprintf(['Something is wrong with the Personalized X Axis. \n' 'Matlab says: \n' ME.message]);
264                 uialert(app.UIFigure, str, 'Sintax Problem');
265             end
266         end
267
268         % Value changed function: PersonalizedYAxisTextArea
269         function PersonalizedYAxisTextAreaValueChanged(app, event)
270             value = app.PersonalizedYAxisTextArea.Value;
271             try
272                 app.condY = str2func(value{:});
273             catch ME
274                 str = sprintf(['Something is wrong with the Personalized Y Axis. \n' 'Matlab says: \n' ME.message]);
275                 uialert(app.UIFigure, str, 'Sintax Problem');
276             end
277         end
278
279         % Value changed function: PersonalizedZAxisTextArea
280         function PersonalizedZAxisTextAreaValueChanged(app, event)
281             value = app.PersonalizedZAxisTextArea.Value;
282             try
283                 app.condZ = str2func(value{:});
284             catch ME
285                 str = sprintf(['Something is wrong with the Personalized Y Axis. \n' 'Matlab says: \n' ME.message]);
286                 uialert(app.UIFigure, str, 'Sintax Problem');
287             end
288         end
289
290         % Value changed function: DataConditionDropDown
291         function DataConditionDropDownValueChanged(app, event)
292             value = app.DataConditionDropDown.Value;
293             app.PersonalizedDataConditionTextArea.Editable = ”off”;
294             switch value
295                 case 'Double Band'
296                     app.condData = app.cDoubleBand;
297                 case 'Center Band'
298                     app.condData = app.cCenterBand;
299                 case 'Both (OR)'
300                     app.condData = app.cBothOR;
301                 case 'Both (AND)'
302                     app.condData = app.cBothAND;
303                 case '4 Resistances'
304                     app.condData = app.c4Rs;
305                 case 'None'
306                     app.condData = app.cNone;
307                 case 'Personalized'
308                     app.PersonalizedDataConditionTextArea.Editable = ”on”;
309                     app.PersonalizedDataConditionTextAreaValueChanged;
```

```
310             end
311             app.PersonalizedDataConditionTextArea.Value = func2str(app.condData);
312         end
313 
314         % Value changed function: PersonalizedDataConditionTextArea
315         function PersonalizedDataConditionTextAreaValueChanged(app, event)
316             value = app.PersonalizedDataConditionTextArea.Value;
317             try
318                 app.condData = str2func(value{:});
319             catch ME
320                 str = sprintf(['Something is wrong with the Personalized Data Condition. \n'↩
                        ↪  'Matlab says: \n' ME.message]);
321                 uialert(app.UIFigure, str, 'Sintax Problem');
322             end
323         end
324 
325         % Button pushed function: ApplyConditionButton
326         function ApplyConditionButtonPushed(app, event)
327             tic
328             d = uiprogressdlg(app.UIFigure, 'Title','Applying Condition', 'Indeterminate',"on↩
                    ↪ ");
329             try
330                 app.ds.reset;
331                 app.tData = parforApplyCondition(app, app.ds, app.condData);
332                 % app.tData = tall(app.ds);
333                 % app.tData = app.tData(app.condData(app.tData),:);
334                 app.ds.reset;
335             catch ME
336                 str = sprintf(['Something is wrong with the datastore loaded. \n' 'Matlab ↩
                        ↪ says: \n' ME.message]);
337                 uialert(app.UIFigure, str, 'Data Problem');
338             end
339 
340             close(d);
341             toc
342         end
343 
344         % Button pushed function: VisualizeButton
345         function VisualizeButtonPushed(app, event)
346             tic
347             d = uiprogressdlg(app.UIFigure, 'Title','Creating Visualization', 'Indeterminate',"↩
                    ↪ on");
348             try
349                 figure,
350                 app.tData.X = app.condX(app.tData);
351                 app.tData.Y = app.condY(app.tData);
352                 if app.plt3d
353                     app.tData.Z = app.condZ(app.tData);
354                     pt1 = scatter3(app.tData.X, app.tData.Y,app.tData.Z,5,app.tData.Z,'x');
355                     m=100;
356                     cm_inferno=inferno();
```

```
357                     colormap(cm_inferno);
358                     hc=colorbar;
359                     title(hc,app.ZAxisDropDown.Value,'Interpreter',"latex");
360                     ini=3;
361                     pt1.DataTipTemplate.DataTipRows(1).Label = app.XAxisDropDown.↩
                            ↪ Value;
362                     pt1.DataTipTemplate.DataTipRows(2).Label = app.YAxisDropDown.↩
                            ↪ Value;
363                     pt1.DataTipTemplate.DataTipRows(3).Label = app.ZAxisDropDown.↩
                            ↪ Value;
364                     xlabel(app.XAxisDropDown.Value);
365                     ylabel(app.YAxisDropDown.Value);
366                     zlabel(app.ZAxisDropDown.Value);
367                     view(0,90)
368                 else
369                     pt1 = plot(app.tData.X, app.tData.Y,'kx','MarkerSize',4);
370                     ini = 2;
371                     pt1.DataTipTemplate.DataTipRows(1).Label = app.XAxisDropDown.↩
                            ↪ Value;
372                     pt1.DataTipTemplate.DataTipRows(2).Label = app.YAxisDropDown.↩
                            ↪ Value;
373                     xlabel(app.XAxisDropDown.Value);
374                     ylabel(app.YAxisDropDown.Value);
375                 end
376                 grid("on")
377                 pt1.DataTipTemplate.DataTipRows(ini+1) = dataTipTextRow('R1',app.↩
                        ↪ tData.R1);
378                 pt1.DataTipTemplate.DataTipRows(ini+2) = dataTipTextRow('Z1',app.↩
                        ↪ tData.Z1);
379                 pt1.DataTipTemplate.DataTipRows(ini+3) = dataTipTextRow('R2',app.↩
                        ↪ tData.R2);
380                 pt1.DataTipTemplate.DataTipRows(ini+4) = dataTipTextRow('R3',app.↩
                        ↪ tData.R3);
381                 pt1.DataTipTemplate.DataTipRows(ini+5) = dataTipTextRow('Z2',app.↩
                        ↪ tData.Z2);
382                 pt1.DataTipTemplate.DataTipRows(ini+6) = dataTipTextRow('R4',app.↩
                        ↪ tData.R4);
383                 % scatter(app.condX(app.tData), app.condY(app.tData),'*')
384                 title(app.TitleEditField.Value)
385                 setfigpaper('Width', 15,'Interpreter','Latex');
386             catch ME
387                 str = sprintf(['Something is wrong with the datastore loaded. \n' 'Matlab ↩
                        ↪ says: \n' ME.message]);
388                 uialert(app.UIFigure, str, 'Data Problem');
389             end
390
391             close(d);
392             toc
393         end
394
395         % Button pushed function: OpenButton
```

```
396         function OpenButtonPushed(app, event)
397             folderpath = uigetdir;
398             d = uiprogressdlg(app.UIFigure, 'Title','Loading Data', 'Indeterminate',"on");
399             try
400                 app.ds = datastore(folderpath);
401                 app.ds.reset;
402                 app.FolderPathEditField.Value = folderpath;
403             catch ME
404                 str = sprintf(['Something is wrong with the datastore loaded. \n' 'Matlab says: \n' ME.message]);
405                 uialert(app.UIFigure, str, 'Load Problem');
406             end
407             close(d);
408         end
409     end
410 
411     % Component initialization
412     methods (Access = private)
413 
414         % Create UIFigure and components
415         function createComponents(app)
416 
417             % Create UIFigure and hide until all components are created
418             app.UIFigure = uifigure('Visible', 'off');
419             app.UIFigure.AutoResizeChildren = 'off';
420             app.UIFigure.Position = [100 100 640 480];
421             app.UIFigure.Name = 'UI Figure';
422 
423             % Create GridLayout
424             app.GridLayout = uigridlayout(app.UIFigure);
425             app.GridLayout.RowHeight = {'1x'};
426             app.GridLayout.ColumnSpacing = 0;
427             app.GridLayout.RowSpacing = 0;
428             app.GridLayout.Padding = [0 0 0 0];
429             app.GridLayout.Scrollable = 'on';
430 
431             % Create LeftPanel
432             app.LeftPanel = uipanel(app.GridLayout);
433             app.LeftPanel.AutoResizeChildren = 'off';
434             app.LeftPanel.Layout.Row = 1;
435             app.LeftPanel.Layout.Column = 1;
436             app.LeftPanel.Scrollable = 'on';
437 
438             % Create OpenButton
439             app.OpenButton = uibutton(app.LeftPanel, 'push');
440             app.OpenButton.ButtonPushedFcn = createCallbackFcn(app, @OpenButtonPushed, true);
441             app.OpenButton.Position = [246 359 40 22];
442             app.OpenButton.Text = 'Open';
443 
444             % Create ApplyConditionButton
```

```
445             app.ApplyConditionButton = uibutton(app.LeftPanel, 'push');
446             app.ApplyConditionButton.ButtonPushedFcn = createCallbackFcn(app, @↩
                    ↪ ApplyConditionButtonPushed, true);
447             app.ApplyConditionButton.Position = [186 179 100 22];
448             app.ApplyConditionButton.Text = 'Apply Condition';
449
450             % Create FolderPathEditFieldLabel
451             app.FolderPathEditFieldLabel = uilabel(app.LeftPanel);
452             app.FolderPathEditFieldLabel.HorizontalAlignment = 'right';
453             app.FolderPathEditFieldLabel.Position = [56 359 67 22];
454             app.FolderPathEditFieldLabel.Text = 'Folder Path';
455
456             % Create FolderPathEditField
457             app.FolderPathEditField = uieditfield(app.LeftPanel, 'text');
458             app.FolderPathEditField.Editable = 'off';
459             app.FolderPathEditField.Position = [136 359 100 22];
460
461             % Create DataConditionDropDownLabel
462             app.DataConditionDropDownLabel = uilabel(app.LeftPanel);
463             app.DataConditionDropDownLabel.HorizontalAlignment = 'right';
464             app.DataConditionDropDownLabel.Position = [36 289 85 22];
465             app.DataConditionDropDownLabel.Text = 'Data Condition';
466
467             % Create DataConditionDropDown
468             app.DataConditionDropDown = uidropdown(app.LeftPanel);
469             app.DataConditionDropDown.ValueChangedFcn = createCallbackFcn(app, @↩
                    ↪ DataConditionDropDownValueChanged, true);
470             app.DataConditionDropDown.Position = [136 289 150 22];
471
472             % Create PersonalizedDataConditionTextAreaLabel
473             app.PersonalizedDataConditionTextAreaLabel = uilabel(app.LeftPanel);
474             app.PersonalizedDataConditionTextAreaLabel.HorizontalAlignment = 'right';
475             app.PersonalizedDataConditionTextAreaLabel.Position = [36 246 85 27];
476             app.PersonalizedDataConditionTextAreaLabel.Text = {'Personalized'; 'Data ↩
                    ↪ Condition'};
477
478             % Create PersonalizedDataConditionTextArea
479             app.PersonalizedDataConditionTextArea = uitextarea(app.LeftPanel);
480             app.PersonalizedDataConditionTextArea.ValueChangedFcn = createCallbackFcn(↩
                    ↪ app, @PersonalizedDataConditionTextAreaValueChanged, true);
481             app.PersonalizedDataConditionTextArea.Editable = 'off';
482             app.PersonalizedDataConditionTextArea.Position = [136 215 150 60];
483
484             % Create TitleEditFieldLabel
485             app.TitleEditFieldLabel = uilabel(app.LeftPanel);
486             app.TitleEditFieldLabel.HorizontalAlignment = 'right';
487             app.TitleEditFieldLabel.Position = [94 111 27 22];
488             app.TitleEditFieldLabel.Text = 'Title';
489
490             % Create TitleEditField
491             app.TitleEditField = uieditfield(app.LeftPanel, 'text');
```

```
492             app.TitleEditField.Position = [136 111 150 22];
493             app.TitleEditField.Value = 'Modelo 1';
494 
495             % Create RightPanel
496             app.RightPanel = uipanel(app.GridLayout);
497             app.RightPanel.AutoResizeChildren = 'off';
498             app.RightPanel.Layout.Row = 1;
499             app.RightPanel.Layout.Column = 2;
500             app.RightPanel.Scrollable = 'on';
501 
502             % Create VisualizeButton
503             app.VisualizeButton = uibutton(app.RightPanel, 'push');
504             app.VisualizeButton.ButtonPushedFcn = createCallbackFcn(app, @↩
                    ↪ VisualizeButtonPushed, true);
505             app.VisualizeButton.Position = [180 77 100 22];
506             app.VisualizeButton.Text = 'Visualize';
507 
508             % Create XAxisDropDownLabel
509             app.XAxisDropDownLabel = uilabel(app.RightPanel);
510             app.XAxisDropDownLabel.HorizontalAlignment = 'right';
511             app.XAxisDropDownLabel.Position = [77 418 39 22];
512             app.XAxisDropDownLabel.Text = 'X Axis';
513 
514             % Create XAxisDropDown
515             app.XAxisDropDown = uidropdown(app.RightPanel);
516             app.XAxisDropDown.ValueChangedFcn = createCallbackFcn(app, @↩
                    ↪ XAxisDropDownValueChanged, true);
517             app.XAxisDropDown.Position = [131 418 150 22];
518 
519             % Create PersonalizedXAxisTextAreaLabel
520             app.PersonalizedXAxisTextAreaLabel = uilabel(app.RightPanel);
521             app.PersonalizedXAxisTextAreaLabel.HorizontalAlignment = 'right';
522             app.PersonalizedXAxisTextAreaLabel.Position = [41 379 75 27];
523             app.PersonalizedXAxisTextAreaLabel.Text = {'Personalized'; 'X Axis'};
524 
525             % Create PersonalizedXAxisTextArea
526             app.PersonalizedXAxisTextArea = uitextarea(app.RightPanel);
527             app.PersonalizedXAxisTextArea.ValueChangedFcn = createCallbackFcn(app, @↩
                    ↪ PersonalizedXAxisTextAreaValueChanged, true);
528             app.PersonalizedXAxisTextArea.Editable = 'off';
529             app.PersonalizedXAxisTextArea.Position = [131 348 150 60];
530 
531             % Create YAxisDropDownLabel
532             app.YAxisDropDownLabel = uilabel(app.RightPanel);
533             app.YAxisDropDownLabel.HorizontalAlignment = 'right';
534             app.YAxisDropDownLabel.Position = [77 306 39 22];
535             app.YAxisDropDownLabel.Text = 'Y Axis';
536 
537             % Create YAxisDropDown
538             app.YAxisDropDown = uidropdown(app.RightPanel);
539             app.YAxisDropDown.ValueChangedFcn = createCallbackFcn(app, @↩
```

```
            ↪ YAxisDropDownValueChanged, true);
540         app.YAxisDropDown.Position = [131 306 150 22];
541
542         % Create PersonalizedYAxisTextAreaLabel
543         app.PersonalizedYAxisTextAreaLabel = uilabel(app.RightPanel);
544         app.PersonalizedYAxisTextAreaLabel.HorizontalAlignment = 'right';
545         app.PersonalizedYAxisTextAreaLabel.Position = [40 272 75 27];
546         app.PersonalizedYAxisTextAreaLabel.Text = {'Personalized'; 'Y Axis'};
547
548         % Create PersonalizedYAxisTextArea
549         app.PersonalizedYAxisTextArea = uitextarea(app.RightPanel);
550         app.PersonalizedYAxisTextArea.ValueChangedFcn = createCallbackFcn(app, @↩
            ↪ PersonalizedYAxisTextAreaValueChanged, true);
551         app.PersonalizedYAxisTextArea.Editable = 'off';
552         app.PersonalizedYAxisTextArea.Position = [130 236 150 60];
553
554         % Create ZAxisDropDownLabel
555         app.ZAxisDropDownLabel = uilabel(app.RightPanel);
556         app.ZAxisDropDownLabel.HorizontalAlignment = 'right';
557         app.ZAxisDropDownLabel.Position = [77 194 39 22];
558         app.ZAxisDropDownLabel.Text = 'Z Axis';
559
560         % Create ZAxisDropDown
561         app.ZAxisDropDown = uidropdown(app.RightPanel);
562         app.ZAxisDropDown.ValueChangedFcn = createCallbackFcn(app, @↩
            ↪ ZAxisDropDownValueChanged, true);
563         app.ZAxisDropDown.Position = [131 194 150 22];
564
565         % Create PersonalizedZAxisTextAreaLabel
566         app.PersonalizedZAxisTextAreaLabel = uilabel(app.RightPanel);
567         app.PersonalizedZAxisTextAreaLabel.HorizontalAlignment = 'right';
568         app.PersonalizedZAxisTextAreaLabel.Position = [40 155 75 27];
569         app.PersonalizedZAxisTextAreaLabel.Text = {'Personalized'; 'Z Axis'};
570
571         % Create PersonalizedZAxisTextArea
572         app.PersonalizedZAxisTextArea = uitextarea(app.RightPanel);
573         app.PersonalizedZAxisTextArea.ValueChangedFcn = createCallbackFcn(app, @↩
            ↪ PersonalizedZAxisTextAreaValueChanged, true);
574         app.PersonalizedZAxisTextArea.Editable = 'off';
575         app.PersonalizedZAxisTextArea.Position = [130 124 150 60];
576
577         % Show the figure after all components are created
578         app.UIFigure.Visible = 'on';
579     end
580 end
581
582 % App creation and deletion
583 methods (Access = public)
584
585     % Construct app
586     function app = visualizerMod1
```

```
587
588             % Create UIFigure and components
589             createComponents(app)
590
591             % Register the app with App Designer
592             registerApp(app, app.UIFigure)
593
594             % Execute the startup function
595             runStartupFcn(app, @startupFcn)
596
597             if nargout == 0
598                 clear app
599             end
600         end
601
602         % Code that executes before app deletion
603         function delete(app)
604
605             % Delete UIFigure when app is deleted
606             delete(app.UIFigure)
607         end
608     end
609 end
```

Código D.7: Función para calcular el histograma en paralelo

```
function [histo] = parforHistograma(ds,cond)
N = numpartitions(ds,gcp);
histo = zeros(21,6);
parfor ii = 1:N
    % Get partition ii of the datastore.
    subds = partition(ds,N,ii);
    hist = zeros(21,6);
    r1=0; z1=0; r2=0; r3=0; z2=0; r4=0;
    R = linspace(-2.5,102.5,22);
    Z = linspace(22.5,127.5,22);
    while (hasdata(subds))
        dataALL = read(subds);
        data = dataALL(cond(dataALL),:);
        r1 = histcounts(data.R1,R);
        z1 = histcounts(data.Z1,Z);
        r2 = histcounts(data.R2,R);
        r3 = histcounts(data.R3,R);
        z2 = histcounts(data.Z2,Z);
        r4 = histcounts(data.R4,R);
        hist(:,1) = hist(:,1) + r1';
        hist(:,2) = hist(:,2) + z1';
        hist(:,3) = hist(:,3) + r2';
        hist(:,4) = hist(:,4) + r3';
        hist(:,5) = hist(:,5) + z2';
        hist(:,6) = hist(:,6) + r4';
```

```
    end
    histo = histo + hist;
end
end
```

Código D.8: Script de análisis (Histograma) del Modelo 1

```
%% Analysis Model 1
addpath('funcs');
%% Carga DataStore y Nombre de Variables
ds = datastore('E:\TFM_DATA\Mod1DS_DoubleBand\');
%% Condiciones Tabla
cNone = @(x) 1:height(x);
%% Análisis
ds.reset;
tic
[Histo] = parforHistograma(ds, cNone);
toc
%% Representación
R = linspace(0,100,21);
Z = linspace(25,125,21);
figure('Name','Double Band'),
subplot(2,3,1), bar(R,Histo(:,1),'b'), title('R1'), xlabel('\Omega')
subplot(2,3,2), bar(Z,Histo(:,2),'b'), title('Z1'), xlabel('\Omega'), xticks(linspace(25,125,6))
subplot(2,3,3), bar(R,Histo(:,3),'b'), title('R2'), xlabel('\Omega')
subplot(2,3,4), bar(R,Histo(:,4),'b'), title('R3'), xlabel('\Omega')
subplot(2,3,5), bar(Z,Histo(:,5),'b'), title('Z2'), xlabel('\Omega'), xticks(linspace(25,125,6))
subplot(2,3,6), bar(R,Histo(:,6),'b'), title('R4'), xlabel('\Omega')
```

Código D.9: Aplicación para generar curvas de diseño para el Modelo 1

```
classdef designChartsMod1 < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure matlab.ui.Figure
        GridLayout matlab.ui.container.GridLayout
        LeftPanel matlab.ui.container.Panel
        OpenButton matlab.ui.control.Button
        FolderPathEditFieldLabel matlab.ui.control.Label
        FolderPathEditField matlab.ui.control.EditField
        ApplyConditionButton matlab.ui.control.Button
        DataConditionDropDownLabel matlab.ui.control.Label
        DataConditionDropDown matlab.ui.control.DropDown
        PersonalizedDataConditionTextAreaLabel matlab.ui.control.Label
        PersonalizedDataConditionTextArea matlab.ui.control.TextArea
        RightPanel matlab.ui.container.Panel
        PlotButton matlab.ui.control.Button
        R1DropDownLabel matlab.ui.control.Label
        R1DropDown matlab.ui.control.DropDown
        Z1DropDownLabel matlab.ui.control.Label
```

```
        Z1DropDown matlab.ui.control.DropDown
        R2DropDownLabel matlab.ui.control.Label
        R2DropDown matlab.ui.control.DropDown
        R3DropDownLabel matlab.ui.control.Label
        R3DropDown matlab.ui.control.DropDown
        Z2DropDownLabel matlab.ui.control.Label
        Z2DropDown matlab.ui.control.DropDown
        R4DropDownLabel matlab.ui.control.Label
        R4DropDown matlab.ui.control.DropDown
        YAxisDropDownLabel matlab.ui.control.Label
        YAxisDropDown matlab.ui.control.DropDown
        XAxisDropDownLabel matlab.ui.control.Label
        XAxisDropDown matlab.ui.control.DropDown
        LegendDropDownLabel matlab.ui.control.Label
        LegendDropDown matlab.ui.control.DropDown
        PersonalizedYAxisLabel matlab.ui.control.Label
        PersonalizedYAxisTextArea matlab.ui.control.TextArea
        PersonalizedXAxisTextAreaLabel matlab.ui.control.Label
        PersonalizedXAxisTextArea matlab.ui.control.TextArea
        TitleEditFieldLabel matlab.ui.control.Label
        TitleEditField matlab.ui.control.EditField
    end


    properties (Access = private)
        onePanelWidth = 576;
        ds % Description
        tData % Description
        yVariable % Description
        xVariable % Description
        legendVariable % Description
        r1=NaN % Description
        z1=NaN % Description
        r2=NaN % Description
        r3=NaN % Description
        z2=NaN % Description
        r4=NaN % Description
        plotData % Description

        cDoubleBand = @(x) (x.Del1<=-1e-9 & x.S211>=0.1);
        cCenterBand = @(x) (x.DelC<=-1e-9 & x.S21C>=0.1);
        cBothOR = @(x) (x.Del1<=-1e-9 & x.S211>=0.1) | (x.DelC<=-1e-9 & x.S21C↩
            ↪ >=0.1);
        cBothAND = @(x) (x.Del1<=-1e-9 & x.S211>=0.1) & (x.DelC<=-1e-9 & x.S21C↩
            ↪ >=0.1);
        cNone = @(x) 1:height(x);
        c4Rs = @(x) (x.R1~=0 & x.R2~=0 & x.R3~=0 & x.R4~=0);

        condData = @(x) 1:height(x);
    end

```

```
70  methods (Access = private)
71
72      % Changes arrangement of the app based on UIFigure width
73      function updateAppLayout(app, event)
74          currentFigureWidth = app.UIFigure.Position(3);
75          if(currentFigureWidth <= app.onePanelWidth)
76              % Change to a 2x1 grid
77              app.GridLayout.RowHeight = {480, 480};
78              app.GridLayout.ColumnWidth = {'1x'};
79              app.RightPanel.Layout.Row = 2;
80              app.RightPanel.Layout.Column = 1;
81          else
82              % Change to a 1x2 grid
83              app.GridLayout.RowHeight = {'1x'};
84              app.GridLayout.ColumnWidth = {'1x', '1x'};
85              app.RightPanel.Layout.Row = 1;
86              app.RightPanel.Layout.Column = 2;
87          end
88      end
89
90
91
92      function [newData] = parforApplyConditions(app, ds)
93          rr1 = app.r1;
94          zz1 = app.z1;
95          rr2 = app.r2;
96          rr3 = app.r3;
97          zz2 = app.z2;
98          rr4 = app.r4;
99          funstr = "@(x)";
100         if ~isnan(rr1)
101             funstr=funstr.append(sprintf("x.R1==%d&",rr1));
102         end
103         if ~isnan(zz1)
104             funstr=funstr.append(sprintf("x.Z1==%d&",zz1));
105         end
106         if ~isnan(rr2)
107             funstr=funstr.append(sprintf("x.R2==%d&",rr2));
108         end
109         if ~isnan(rr3)
110             funstr=funstr.append(sprintf("x.R3==%d&",rr3));
111         end
112         if ~isnan(zz2)
113             funstr=funstr.append(sprintf("x.Z2==%d&",zz2));
114         end
115         if ~isnan(rr4)
116             funstr=funstr.append(sprintf("x.R4==%d&",rr4));
117         end
118         funchar = char(funstr);
119         funstr = string(funchar(1:end-1));
120         cond = str2func(funstr);
```

```
121             clear funchar funstr rr1 rr2 rr3 rr4 zz1 zz2
122             N = numpartitions(ds,gcp);
123             newData = [];
124             parfor ii = 1:N
125                 subds = partition(ds,N,ii);
126                 Data = [];
127                 while hasdata(subds)
128                     data = read(subds);
129                     data = data(cond(data),:)
130                     Data = [Data; data];
131                 end
132                 newData = [newData; Data];
133             end
134         end
135
136         function [newData] = parforApplyCondition(app, ds,cond)
137             N = numpartitions(ds,gcp);
138             newData = [];
139             parfor ii = 1:N
140                 subds = partition(ds,N,ii);
141                 Data = [];
142                 while hasdata(subds)
143                     data = read(subds);
144                     data = data(cond(data),:);
145                     Data = [Data; data];
146                 end
147                 newData = [newData; Data];
148             end
149         end
150     end
151
152
153     % Callbacks that handle component events
154     methods (Access = private)
155
156         % Code that executes after component creation
157         function startupFcn(app)
158             app.UIFigure.SizeChangedFcn = createCallbackFcn(app, @updateAppLayout, ↩
                    ↪ true);
159             % app.ds = datastore('E:\TFM_DATA\Mod1DS\');
160             % app.ds.reset;
161             app.YAxisDropDown.Items = {'NGD f1', 'NGD fc', 'S21 f1', 'S21 fc', 'BW f1', '↩
                    ↪ BW fc', 'f2/f1', 'S11 f1', 'S11 fc', 'S22 f1', 'S22 fc','Personalized'};
162             app.XAxisDropDown.Items = {'R1', 'Z1', 'R2', 'R3', 'Z2', 'R4', 'Personalized'};
163             app.LegendDropDown.Items = {'R1', 'Z1', 'R2', 'R3', 'Z2', 'R4'};
164             app.LegendDropDown.Value = 'Z1';
165             app.DataConditionDropDown.Items = {'None','Double Band', 'Center Band', '↩
                    ↪ Both (OR)', 'Both (AND)', '4 Resistances', 'Personalized'};
166             app.PersonalizedDataConditionTextArea.Value = func2str(app.condData);
167             app.R1DropDown.Items = {'NaN', '0', '5', '10', '15', '20', '25', '30', '35', '40', '45'↩
                    ↪ , '50', '55', '60', '65', '70', '75', '80', '85', '90', '95', '100'};
```

```
168             app.Z1DropDown.Items = {'NaN','25', '30', '35', '40', '45', '50', '55', '60', '65', '↩
                    ↪ 70', '75', '80', '85', '90', '95', '100', '105', '110', '115', '120', '125'};
169             app.R2DropDown.Items = {'NaN', '0', '5', '10', '15', '20', '25', '30', '35', '40', '45'↩
                    ↪ , '50', '55', '60', '65', '70', '75', '80', '85', '90', '95', '100'};
170             app.R3DropDown.Items = {'NaN', '0', '5', '10', '15', '20', '25', '30', '35', '40', '45'↩
                    ↪ , '50', '55', '60', '65', '70', '75', '80', '85', '90', '95', '100'};
171             app.Z2DropDown.Items = {'NaN','25', '30', '35', '40', '45', '50', '55', '60', '65', '↩
                    ↪ 70', '75', '80', '85', '90', '95', '100', '105', '110', '115', '120', '125'};
172             app.R4DropDown.Items = {'NaN', '0', '5', '10', '15', '20', '25', '30', '35', '40', '45'↩
                    ↪ , '50', '55', '60', '65', '70', '75', '80', '85', '90', '95', '100'};
173             % app.R1DropDown.Enable = 'off';
174             % app.Z1DropDown.Enable = 'off';
175             app.xVariable = app.XAxisDropDown.Value;
176             app.legendVariable = app.LegendDropDown.Value;
177             app.yVariable = @(x) x.Del1;
178             app.xVariable = ”x.R1”;
179             app.PersonalizedYAxisTextArea.Value = func2str(app.yVariable);
180             app.PersonalizedXAxisTextArea.Value = app.xVariable;
181             estilo = {'-+','-o','-*','-x','-s','-d','-p'};
182             es = repmat(estilo,[3,1]);
183             es = reshape(es.',1,[]);
184             set(groot,'defaultAxesLineStyleOrder',es)
185 
186         end
187 
188         % Value changed function: XAxisDropDown
189         function XAxisDropDownValueChanged(app, event)
190             value = app.XAxisDropDown.Value;
191             app.PersonalizedXAxisTextArea.Editable = 'off';
192             switch value
193                 case 'Personalized'
194                     app.PersonalizedXAxisTextArea.Editable = ”on”;
195                     app.PersonalizedXAxisTextAreaValueChanged;
196                 otherwise
197                     app.xVariable = ”x.”+string(value);
198             end
199             app.PersonalizedXAxisTextArea.Value = app.xVariable;
200         end
201 
202         % Value changed function: LegendDropDown
203         function LegendDropDownValueChanged(app, event)
204             value = app.LegendDropDown.Value;
205             switch value
206                 case 'R1'
207                     app.R1DropDown.Value = 'NaN';
208                     app.r1 = NaN;
209                     app.R1DropDown.Enable = 'off';
210                 case 'Z1'
211                     app.Z1DropDown.Value = 'NaN';
212                     app.z1 = NaN;
213                     app.Z1DropDown.Enable = 'off';
```

```
214                 case 'R2'
215                     app.R2DropDown.Value = 'NaN';
216                     app.r2 = NaN;
217                     app.R2DropDown.Enable = 'off';
218                 case 'R3'
219                     app.R3DropDown.Value = 'NaN';
220                     app.r3 = NaN;
221                     app.R3DropDown.Enable = 'off';
222                 case 'Z2'
223                     app.Z2DropDown.Value = 'NaN';
224                     app.z2 = NaN;
225                     app.Z2DropDown.Enable = 'off';
226                 case 'R4'
227                     app.R4DropDown.Value = 'NaN';
228                     app.r4 = NaN;
229                     app.R4DropDown.Enable = 'off';
230             end
231             switch app.legendVariable
232                 case 'R1'
233                     app.R1DropDown.Enable = 'on';
234                     app.legendVariable = value;
235                 case 'Z1'
236                     app.Z1DropDown.Enable = 'on';
237                     app.legendVariable = value;
238                 case 'R2'
239                     app.R2DropDown.Enable = 'on';
240                     app.legendVariable = value;
241                 case 'R3'
242                     app.R3DropDown.Enable = 'on';
243                     app.legendVariable = value;
244                 case 'Z2'
245                     app.Z2DropDown.Enable = 'on';
246                     app.legendVariable = value;
247                 case 'R4'
248                     app.R4DropDown.Enable = 'on';
249                     app.legendVariable = value;
250             end
251         end
252 
253         % Value changed function: YAxisDropDown
254         function YAxisDropDownValueChanged(app, event)
255             value = app.YAxisDropDown.Value;
256             app.PersonalizedYAxisTextArea.Editable = 'off';
257 
258             switch value
259                 case 'NGD f1'
260                     app.yVariable = @(x) x.Del1;
261                 case 'NGD fc'
262                     app.yVariable = @(x) x.DelC;
263                 case 'S21 f1'
264                     app.yVariable = @(x) 20*log10(x.S211);
```

```
265                 case 'S21 fc'
266                     app.yVariable = @(x) 20*log10(x.S21C);
267                 case 'BW f1'
268                     app.yVariable = @(x) x.iBW12 - x.iBW11;
269                 case 'BW fc'
270                     app.yVariable = @(x) x.iBW1C - x.iBW1C;
271                 case 'f2/f1'
272                     app.yVariable = @(x) (2000-x.iDel1)./x.iDel1;
273                 case 'S11 f1'
274                     app.yVariable = @(x) 20*log10(x.S111);
275                 case 'S11 fc'
276                     app.yVariable = @(x) 20*log10(x.S11C);
277                 case 'S22 f1'
278                     app.yVariable = @(x) 20*log10(x.S221);
279                 case 'S22 fc'
280                     app.yVariable = @(x) 20*log10(x.S22C);
281                 case 'Personalized'
282                     app.PersonalizedYAxisTextArea.Editable = "on";
283                     app.PersonalizedYAxisTextAreaValueChanged;
284             end
285             app.PersonalizedYAxisTextArea.Value = func2str(app.yVariable);
286         end
287 
288         % Button pushed function: PlotButton
289         function PlotButtonPushed(app, event)
290             tic
291             d = uiprogressdlg(app.UIFigure, 'Title','Applying Condition', 'Indeterminate',"on↩
                    ↪ ");
292             try
293                 app.ds.reset;
294                 app.tData = parforApplyConditions(app, app.ds);
295                 app.ds.reset;
296             catch ME
297                 str = sprintf(['Something is wrong with the datastore loaded. \n' 'Matlab ↩
                        ↪ says: \n' ME.message]);
298                 uialert(app.UIFigure, str, 'Data Problem');
299             end
300 
301             close(d);
302             toc
303             fstr = "@(x)x."+string(app.legendVariable);
304             xCond = str2func("@(x)"+string(app.xVariable));
305             condLegend = str2func(fstr);
306             uniqLegend = unique(condLegend(app.tData));
307             plt = table();
308             fig=figure,
309             hold on;
310             leyenda = {};
311 
312             for ii=1:length(uniqLegend)
313                 aux = fstr.append(sprintf("==%d",uniqLegend(ii)));
```

```
314                 cond = str2func(aux);
315                 plt.Y = app.yVariable(app.tData(cond(app.tData),:));
316                 plt.X = xCond(app.tData(cond(app.tData),:));
317                 pt1 = plot(plt.X,plt.Y);
318                 aux = char(aux);
319                 leyenda = {leyenda{:},string(aux(7:end))};
320                 pt1.DataTipTemplate.DataTipRows(1).Label = app.XAxisDropDown.Value;
321                 pt1.DataTipTemplate.DataTipRows(2).Label = app.YAxisDropDown.Value;
322                 pt1.DataTipTemplate.DataTipRows(end+1) = dataTipTextRow(app.↩
                        ↪ LegendDropDown.Value,repmat(uniqLegend(ii),size(plt.X)));
323                 clearvars plt
324             end
325             leg=legend(leyenda,'Location','bestoutside'),
326             grid("on"),
327             xlabel(app.XAxisDropDown.Value);
328             ylabel(app.YAxisDropDown.Value);
329             title(app.TitleEditField.Value)
330             ann=annotation('textbox', [0, 0.5, 0, 0], 'string', sprintf("R1=%d\nZ1=%d\nR2↩
                    ↪ =%d\nR3=%d\nZ2=%d\nR4=%d",app.r1,app.z1,app.r2,app.r3,app.z2,↩
                    ↪ app.r4),'EdgeColor','k','BackgroundColor','w','Position',[leg.Position(1),↩
                    ↪ leg.Position(2)-0.1,0,0],'FitBoxToText',1);
331             setfigpaper('Width',22,'Interpreter','Latex');
332             ann.Position = [leg.Position(1),leg.Position(2)-ann.Position(4)-0.01,ann.Position↩
                    ↪ (3),ann.Position(4)];
333             % movegui(fig,"center")
334         end
335
336         % Value changed function: R1DropDown
337         function R1DropDownValueChanged(app, event)
338             app.r1 = str2double(app.R1DropDown.Value);
339         end
340
341         % Value changed function: Z1DropDown
342         function Z1DropDownValueChanged(app, event)
343             app.z1 = str2double(app.Z1DropDown.Value);
344         end
345
346         % Value changed function: R2DropDown
347         function R2DropDownValueChanged(app, event)
348             app.r2 = str2double(app.R2DropDown.Value);
349         end
350
351         % Value changed function: R3DropDown
352         function R3DropDownValueChanged(app, event)
353             app.r3 = str2double(app.R3DropDown.Value);
354         end
355
356         % Value changed function: Z2DropDown
357         function Z2DropDownValueChanged(app, event)
358             app.z2 = str2double(app.Z2DropDown.Value);
359         end
```

```
360
361         % Value changed function: R4DropDown
362         function R4DropDownValueChanged(app, event)
363             app.r4 = str2double(app.R4DropDown.Value);
364         end
365
366         % Value changed function: PersonalizedYAxisTextArea
367         function PersonalizedYAxisTextAreaValueChanged(app, event)
368             value = app.PersonalizedYAxisTextArea.Value;
369             try
370                 app.yVariable = str2func(value{:});
371             catch ME
372                 str = sprintf(['Something is wrong with the Personalized Y Axis. \n' 'Matlab↩
                        ↪  says: \n' ME.message]);
373                 uialert(app.UIFigure, str, 'Sintax Problem');
374             end
375         end
376
377         % Value changed function: PersonalizedXAxisTextArea
378         function PersonalizedXAxisTextAreaValueChanged(app, event)
379             value = app.PersonalizedXAxisTextArea.Value;
380             try
381                 app.xVariable = string(value{:});
382             catch ME
383                 str = sprintf(['Something is wrong with the Personalized Y Axis. \n' 'Matlab↩
                        ↪  says: \n' ME.message]);
384                 uialert(app.UIFigure, str, 'Sintax Problem');
385             end
386         end
387
388         % Button pushed function: OpenButton
389         function OpenButtonPushed(app, event)
390             folderpath = uigetdir;
391             d = uiprogressdlg(app.UIFigure, 'Title','Loading Data', 'Indeterminate',"on");
392             try
393                 app.ds = datastore(folderpath);
394                 app.ds.reset;
395                 app.FolderPathEditField.Value = folderpath;
396             catch ME
397                 str = sprintf(['Something is wrong with the datastore loaded. \n' 'Matlab ↩
                        ↪ says: \n' ME.message]);
398                 uialert(app.UIFigure, str, 'Load Problem');
399             end
400             close(d);
401         end
402
403         % Button pushed function: ApplyConditionButton
404         function ApplyConditionButtonPushed(app, event)
405             tic
406             d = uiprogressdlg(app.UIFigure, 'Title','Applying Condition', 'Indeterminate',"on↩
                    ↪ ");
```

```
407             try
408                 app.ds.reset;
409                 app.tData = parforApplyCondition(app, app.ds, app.condData);
410                 % app.tData = tall(app.ds);
411                 % app.tData = app.tData(app.condData(app.tData),:);
412                 app.ds.reset;
413             catch ME
414                 str = sprintf(['Something is wrong with the datastore loaded. \n' 'Matlab ↩
                        ↪ says: \n' ME.message]);
415                 uialert(app.UIFigure, str, 'Data Problem');
416             end
417 
418             close(d);
419             toc
420         end
421     end
422 
423     % Component initialization
424     methods (Access = private)
425 
426         % Create UIFigure and components
427         function createComponents(app)
428 
429             % Create UIFigure and hide until all components are created
430             app.UIFigure = uifigure('Visible', 'off');
431             app.UIFigure.AutoResizeChildren = 'off';
432             app.UIFigure.Position = [100 100 640 486];
433             app.UIFigure.Name = 'UI Figure';
434 
435             % Create GridLayout
436             app.GridLayout = uigridlayout(app.UIFigure);
437             app.GridLayout.RowHeight = {'1x'};
438             app.GridLayout.ColumnSpacing = 0;
439             app.GridLayout.RowSpacing = 0;
440             app.GridLayout.Padding = [0 0 0 0];
441             app.GridLayout.Scrollable = 'on';
442 
443             % Create LeftPanel
444             app.LeftPanel = uipanel(app.GridLayout);
445             app.LeftPanel.AutoResizeChildren = 'off';
446             app.LeftPanel.Layout.Row = 1;
447             app.LeftPanel.Layout.Column = 1;
448 
449             % Create OpenButton
450             app.OpenButton = uibutton(app.LeftPanel, 'push');
451             app.OpenButton.ButtonPushedFcn = createCallbackFcn(app, @↩
                    ↪ OpenButtonPushed, true);
452             app.OpenButton.Position = [249 368 40 22];
453             app.OpenButton.Text = 'Open';
454 
455             % Create FolderPathEditFieldLabel
```

```
456             app.FolderPathEditFieldLabel = uilabel(app.LeftPanel);
457             app.FolderPathEditFieldLabel.HorizontalAlignment = 'right';
458             app.FolderPathEditFieldLabel.Position = [48 368 67 22];
459             app.FolderPathEditFieldLabel.Text = 'Folder Path';
460
461             % Create FolderPathEditField
462             app.FolderPathEditField = uieditfield(app.LeftPanel, 'text');
463             app.FolderPathEditField.Editable = 'off';
464             app.FolderPathEditField.Position = [139 368 100 22];
465
466             % Create ApplyConditionButton
467             app.ApplyConditionButton = uibutton(app.LeftPanel, 'push');
468             app.ApplyConditionButton.ButtonPushedFcn = createCallbackFcn(app, @↩
                    ↪ ApplyConditionButtonPushed, true);
469             app.ApplyConditionButton.Position = [189 123 100 22];
470             app.ApplyConditionButton.Text = 'Apply Condition';
471
472             % Create DataConditionDropDownLabel
473             app.DataConditionDropDownLabel = uilabel(app.LeftPanel);
474             app.DataConditionDropDownLabel.HorizontalAlignment = 'right';
475             app.DataConditionDropDownLabel.Position = [30 287 85 22];
476             app.DataConditionDropDownLabel.Text = 'Data Condition';
477
478             % Create DataConditionDropDown
479             app.DataConditionDropDown = uidropdown(app.LeftPanel);
480             app.DataConditionDropDown.Position = [139 287 150 22];
481
482             % Create PersonalizedDataConditionTextAreaLabel
483             app.PersonalizedDataConditionTextAreaLabel = uilabel(app.LeftPanel);
484             app.PersonalizedDataConditionTextAreaLabel.HorizontalAlignment = 'right';
485             app.PersonalizedDataConditionTextAreaLabel.Position = [30 244 85 27];
486             app.PersonalizedDataConditionTextAreaLabel.Text = {'Personalized'; 'Data ↩
                    ↪ Condition'};
487
488             % Create PersonalizedDataConditionTextArea
489             app.PersonalizedDataConditionTextArea = uitextarea(app.LeftPanel);
490             app.PersonalizedDataConditionTextArea.Editable = 'off';
491             app.PersonalizedDataConditionTextArea.Position = [139 213 150 60];
492
493             % Create RightPanel
494             app.RightPanel = uipanel(app.GridLayout);
495             app.RightPanel.AutoResizeChildren = 'off';
496             app.RightPanel.Layout.Row = 1;
497             app.RightPanel.Layout.Column = 2;
498
499             % Create PlotButton
500             app.PlotButton = uibutton(app.RightPanel, 'push');
501             app.PlotButton.ButtonPushedFcn = createCallbackFcn(app, @PlotButtonPushed↩
                    ↪ , true);
502             app.PlotButton.Position = [175 24 100 22];
503             app.PlotButton.Text = 'Plot!';
```

```
504
505             % Create R1DropDownLabel
506             app.R1DropDownLabel = uilabel(app.RightPanel);
507             app.R1DropDownLabel.HorizontalAlignment = 'right';
508             app.R1DropDownLabel.Position = [57 213 25 22];
509             app.R1DropDownLabel.Text = 'R1';
510
511             % Create R1DropDown
512             app.R1DropDown = uidropdown(app.RightPanel);
513             app.R1DropDown.Items = {'NaN'};
514             app.R1DropDown.ValueChangedFcn = createCallbackFcn(app, @↩
                    ↪ R1DropDownValueChanged, true);
515             app.R1DropDown.Position = [97 213 65 22];
516             app.R1DropDown.Value = 'NaN';
517
518             % Create Z1DropDownLabel
519             app.Z1DropDownLabel = uilabel(app.RightPanel);
520             app.Z1DropDownLabel.HorizontalAlignment = 'right';
521             app.Z1DropDownLabel.Position = [57 129 25 22];
522             app.Z1DropDownLabel.Text = 'Z1';
523
524             % Create Z1DropDown
525             app.Z1DropDown = uidropdown(app.RightPanel);
526             app.Z1DropDown.Items = {'NaN'};
527             app.Z1DropDown.ValueChangedFcn = createCallbackFcn(app, @↩
                    ↪ Z1DropDownValueChanged, true);
528             app.Z1DropDown.Position = [97 129 65 22];
529             app.Z1DropDown.Value = 'NaN';
530
531             % Create R2DropDownLabel
532             app.R2DropDownLabel = uilabel(app.RightPanel);
533             app.R2DropDownLabel.HorizontalAlignment = 'right';
534             app.R2DropDownLabel.Position = [57 171 25 22];
535             app.R2DropDownLabel.Text = 'R2';
536
537             % Create R2DropDown
538             app.R2DropDown = uidropdown(app.RightPanel);
539             app.R2DropDown.Items = {'NaN'};
540             app.R2DropDown.ValueChangedFcn = createCallbackFcn(app, @↩
                    ↪ R2DropDownValueChanged, true);
541             app.R2DropDown.Position = [97 171 65 22];
542             app.R2DropDown.Value = 'NaN';
543
544             % Create R3DropDownLabel
545             app.R3DropDownLabel = uilabel(app.RightPanel);
546             app.R3DropDownLabel.HorizontalAlignment = 'right';
547             app.R3DropDownLabel.Position = [170 213 25 22];
548             app.R3DropDownLabel.Text = 'R3';
549
550             % Create R3DropDown
551             app.R3DropDown = uidropdown(app.RightPanel);
```

```
552             app.R3DropDown.Items = {'NaN'};
553             app.R3DropDown.ValueChangedFcn = createCallbackFcn(app, @↩
                    ↪ R3DropDownValueChanged, true);
554             app.R3DropDown.Position = [210 213 65 22];
555             app.R3DropDown.Value = 'NaN';
556 
557             % Create Z2DropDownLabel
558             app.Z2DropDownLabel = uilabel(app.RightPanel);
559             app.Z2DropDownLabel.HorizontalAlignment = 'right';
560             app.Z2DropDownLabel.Position = [170 171 25 22];
561             app.Z2DropDownLabel.Text = 'Z2';
562 
563             % Create Z2DropDown
564             app.Z2DropDown = uidropdown(app.RightPanel);
565             app.Z2DropDown.Items = {'NaN'};
566             app.Z2DropDown.ValueChangedFcn = createCallbackFcn(app, @↩
                    ↪ Z2DropDownValueChanged, true);
567             app.Z2DropDown.Position = [210 171 65 22];
568             app.Z2DropDown.Value = 'NaN';
569 
570             % Create R4DropDownLabel
571             app.R4DropDownLabel = uilabel(app.RightPanel);
572             app.R4DropDownLabel.HorizontalAlignment = 'right';
573             app.R4DropDownLabel.Position = [170 129 25 22];
574             app.R4DropDownLabel.Text = 'R4';
575 
576             % Create R4DropDown
577             app.R4DropDown = uidropdown(app.RightPanel);
578             app.R4DropDown.Items = {'NaN'};
579             app.R4DropDown.ValueChangedFcn = createCallbackFcn(app, @↩
                    ↪ R4DropDownValueChanged, true);
580             app.R4DropDown.Position = [210 129 65 22];
581             app.R4DropDown.Value = 'NaN';
582 
583             % Create YAxisDropDownLabel
584             app.YAxisDropDownLabel = uilabel(app.RightPanel);
585             app.YAxisDropDownLabel.HorizontalAlignment = 'right';
586             app.YAxisDropDownLabel.Position = [62 414 39 22];
587             app.YAxisDropDownLabel.Text = 'Y Axis';
588 
589             % Create YAxisDropDown
590             app.YAxisDropDown = uidropdown(app.RightPanel);
591             app.YAxisDropDown.ValueChangedFcn = createCallbackFcn(app, @↩
                    ↪ YAxisDropDownValueChanged, true);
592             app.YAxisDropDown.Position = [125 414 150 22];
593 
594             % Create XAxisDropDownLabel
595             app.XAxisDropDownLabel = uilabel(app.RightPanel);
596             app.XAxisDropDownLabel.HorizontalAlignment = 'right';
597             app.XAxisDropDownLabel.Position = [62 336 39 22];
598             app.XAxisDropDownLabel.Text = 'X Axis';
```

```
599
600         % Create XAxisDropDown
601         app.XAxisDropDown = uidropdown(app.RightPanel);
602         app.XAxisDropDown.ValueChangedFcn = createCallbackFcn(app, @↩
                ↪ XAxisDropDownValueChanged, true);
603         app.XAxisDropDown.Position = [125 336 150 22];
604
605         % Create LegendDropDownLabel
606         app.LegendDropDownLabel = uilabel(app.RightPanel);
607         app.LegendDropDownLabel.HorizontalAlignment = 'right';
608         app.LegendDropDownLabel.Position = [55 260 46 22];
609         app.LegendDropDownLabel.Text = 'Legend';
610
611         % Create LegendDropDown
612         app.LegendDropDown = uidropdown(app.RightPanel);
613         app.LegendDropDown.ValueChangedFcn = createCallbackFcn(app, @↩
                ↪ LegendDropDownValueChanged, true);
614         app.LegendDropDown.Position = [125 260 150 22];
615
616         % Create PersonalizedYAxisLabel
617         app.PersonalizedYAxisLabel = uilabel(app.RightPanel);
618         app.PersonalizedYAxisLabel.HorizontalAlignment = 'right';
619         app.PersonalizedYAxisLabel.Position = [26 380 75 27];
620         app.PersonalizedYAxisLabel.Text = {'Personalized'; 'Y Axis'};
621
622         % Create PersonalizedYAxisTextArea
623         app.PersonalizedYAxisTextArea = uitextarea(app.RightPanel);
624         app.PersonalizedYAxisTextArea.ValueChangedFcn = createCallbackFcn(app, @↩
                ↪ PersonalizedYAxisTextAreaValueChanged, true);
625         app.PersonalizedYAxisTextArea.Position = [125 376 150 33];
626
627         % Create PersonalizedXAxisTextAreaLabel
628         app.PersonalizedXAxisTextAreaLabel = uilabel(app.RightPanel);
629         app.PersonalizedXAxisTextAreaLabel.HorizontalAlignment = 'right';
630         app.PersonalizedXAxisTextAreaLabel.Position = [26 298 75 27];
631         app.PersonalizedXAxisTextAreaLabel.Text = {'Personalized'; 'X Axis'};
632
633         % Create PersonalizedXAxisTextArea
634         app.PersonalizedXAxisTextArea = uitextarea(app.RightPanel);
635         app.PersonalizedXAxisTextArea.ValueChangedFcn = createCallbackFcn(app, @↩
                ↪ PersonalizedXAxisTextAreaValueChanged, true);
636         app.PersonalizedXAxisTextArea.Position = [125 294 150 33];
637
638         % Create TitleEditFieldLabel
639         app.TitleEditFieldLabel = uilabel(app.RightPanel);
640         app.TitleEditFieldLabel.HorizontalAlignment = 'right';
641         app.TitleEditFieldLabel.Position = [74 73 27 22];
642         app.TitleEditFieldLabel.Text = 'Title';
643
644         % Create TitleEditField
645         app.TitleEditField = uieditfield(app.RightPanel, 'text');
```

```
646             app.TitleEditField.Position = [125 73 150 22];
647             app.TitleEditField.Value = 'Modelo 1';
648 
649             % Show the figure after all components are created
650             app.UIFigure.Visible = 'on';
651         end
652     end
653 
654     % App creation and deletion
655     methods (Access = public)
656 
657         % Construct app
658         function app = designChartsMod1
659 
660             % Create UIFigure and components
661             createComponents(app)
662 
663             % Register the app with App Designer
664             registerApp(app, app.UIFigure)
665 
666             % Execute the startup function
667             runStartupFcn(app, @startupFcn)
668 
669             if nargout == 0
670                 clear app
671             end
672         end
673 
674         % Code that executes before app deletion
675         function delete(app)
676 
677             % Delete UIFigure when app is deleted
678             delete(app.UIFigure)
679         end
680     end
681 end
```

E. Códigos Python

Código E.1: Script para el cálculo del Modelo 1

```
import numpy as np
import myFunctions as my
from datetime import datetime
import os
import time
import itertools as it
import multiprocessing
import scipy.io as sc

foldername = "Mod1"
lenRZ = 21
nZeros = 2

R1 = np.linspace(0, 100, num=lenRZ)
R2 = np.linspace(0, 100, num=lenRZ)
R3 = np.linspace(0, 100, num=lenRZ)
R4 = np.linspace(0, 100, num=lenRZ)

Z1 = np.linspace(25, 125, num=lenRZ)
Z2 = np.linspace(25, 125, num=lenRZ)

lenfn = 2001
fn = np.linspace(0, 2, num=lenfn)
if nZeros == 1:
    fn = fn[0:1000]
    lenfn = fn.size
elif nZeros == 2:
    fn = fn[0:1201]
    lenfn = fn.size
# 0 - 999 -->> 1 zero
# 900 - 1200 -->> 2 zeros
# 1001 - end -->> 3 zeros

z0 = 50
f0 = 1e9

f = f0 * fn

xRange = range(lenRZ)


def calcular(counter):
    t = time.time()
    DATA = list()
    ERRORES = list()
    m = np.array(counter)[0]
    n = np.array(counter)[1]
    # for m in x21: # R1
    # for n in x21: # Z1
    for o in xRange: # R2
        for p in xRange: # R3
```

```
52              for q in xRange: # Z2
53                  for r in xRange: # R4
54                      SS11, SS12, SS21, SS22 = my.fCalcSPMod1(R1[m], Z1[n], R2[o], R3[p], Z2[q], R4[r], z0, fn)
55                      group_delay = -(np.diff(my.unwrap(np.angle(SS21), np.pi)) / np.diff(f)) / (2 * np.pi)
56                      iDel, Del, S21, S11, S22, iBW, iBW1dB, iBW3dB, err = my.getMeasures(nZeros, ↩
                            ↪ group_delay, SS21, SS11,
57                                                                                  SS22)
58                      if any(err):
59                          errores = np.concatenate(([R1[m]], [Z1[n]], [R2[o]], [R3[p]], [Z2[q]], [R4[r]], err))
60                          ERRORES.append(errores)
61
62                      datos = np.concatenate(([R1[m]], [Z1[n]], [R2[o]], [R3[p]], [Z2[q]], [R4[r]],
63                                              iDel, Del, S21, S11, S22, iBW3dB, iBW1dB, iBW))
64                      DATA.append(datos)
65                  # end r -> R4
66              # end q -> Z2
67          # end p -> R3
68      # end o -> R2
69      # end n -> Z1
70      # end m -> R1
71      elapsed = time.time() - t
72      t2 = time.time()
73      filename = str(int(R1[m])).zfill(3) + "R1" + str(
74          int(Z1[n])).zfill(3) + "Z1.mat"
75      sc.savemat(foldername + "/" + filename, {'data': DATA}, do_compression=True)
76      sc.savemat(foldername + "_errors" + "/" + "errors_" + filename, {'errors': ERRORES}, do_compression↩
            ↪ =True)
77      elapsed2 = time.time() - t2
78      elapsed3 = time.time() - t
79      print("Archivo " + filename + " creado." +
80            " [" + datetime.now().strftime('%H:%M:%S.%f')[:-4] + "] | Bucle = " + str(elapsed) + " Guardado ↩
                ↪ = " + str(
81          elapsed2) + " Total = " + str(elapsed3))
82      return
83
84
85  if __name__ == "__main__":
86
87      print("Comienzo de ejecución. [" +
88            datetime.now().strftime('%H:%M:%S.%f')[:-4] + "]")
89      aux = list()
90      if not os.path.exists(foldername + "_errors"):
91          os.makedirs(foldername + "_errors")
92      if not os.path.exists(foldername):
93          os.makedirs(foldername)
94      else:
95          files = os.listdir(foldername)
96          for file in files:
97              r1 = int(file[0:3])
98              z1 = int(file[5:8])
99              aux.append((int(np.where(R1 == r1)[0]), int(np.where(Z1 == z1)[0])))
100     t = time.time()
101     M = list(xRange)
102     N = list(xRange)
103     contador = list(it.product(M, N))
104     for elem in aux:
105         contador.remove(elem)
106     # end
107     print(str(np.shape(contador)[0]) + " casos a computar con " + str(multiprocessing.cpu_count()) + " nú↩
            ↪ cleos.")
108     p = multiprocessing.Pool(multiprocessing.cpu_count())
109     p.map(calcular, contador)
110     p.close()
111     p.join()
```

```
    elapsed = time.time() - t
    print("Proceso completo! [" +
          datetime.now().strftime('%H:%M:%S.%f')[:-4] + "]")
    units = " (s)"
    if elapsed > 60:
        elapsed = elapsed / 60
        units = " (min)"
    if elapsed > 60:
        elapsed = elapsed / 60
        units = " (horas)"
    print("Duración total: " + str(elapsed) + units)
```

Código E.2: Función de cálculo de los parámetros S del Modelo 1

```
import numpy as np
import warnings
def fCalcSPMod1(r1, z1, r2, r3, z2, r4, z0, fn):
    warnings.filterwarnings("ignore")
    y1 = 1 / z1 if (z1 != 0) else 0
    y2 = 1 / z2 if (z2 != 0) else 0

    y0 = 1 / z0 if (z0 != 0) else 0
    lenfn = fn.size

    THETA1A = np.pi * fn
    THETA2A = 2 * np.pi * fn
    R1 = np.array([[1, r1], [0, 1]])
    R2 = np.array([[1, r2], [0, 1]])
    R3 = np.array([[1, r3], [0, 1]])
    R4 = np.array([[1, r4], [0, 1]])
    R1 = np.tile(R1, (lenfn, 1, 1))
    R2 = np.tile(R2, (lenfn, 1, 1))
    R3 = np.tile(R3, (lenfn, 1, 1))
    R4 = np.tile(R4, (lenfn, 1, 1))
    TL1 = np.stack([np.stack([np.cos(THETA1A), 1j * z1 * np.sin(THETA1A)], axis=-1),
                    np.stack([1j * y1 * np.sin(THETA1A), np.cos(THETA1A)], axis=-1)], axis=1)
    TL2 = np.stack([np.stack([np.cos(THETA2A), 1j * z2 * np.sin(THETA2A)], axis=-1),
                    np.stack([1j * y2 * np.sin(THETA2A), np.cos(THETA2A)], axis=-1)], axis=1)
    PATH1 = R1 @ TL1 @ R2
    PATH2 = R3 @ TL2 @ R4
    warnings.filterwarnings("default")
    return fPath2SP(PATH1, PATH2, y0)
```

Código E.3: Función para el cálculo de la matriz SP equivalente de una red de dos ramas en paralelo

```
import numpy as np
import warnings
def fPath2SP(PATH1, PATH2, y0):
    warnings.filterwarnings("ignore")
    A1 = PATH1[:, 0, 0]
    B1 = PATH1[:, 0, 1]
    C1 = PATH1[:, 1, 0]
    D1 = PATH1[:, 1, 1]
    A2 = PATH2[:, 0, 0]
    B2 = PATH2[:, 0, 1]
    C2 = PATH2[:, 1, 0]
    D2 = PATH2[:, 1, 1]
    Y_path1 = np.stack(
        [np.stack([(D1 / B1), ((B1 * C1 - A1 * D1) / (B1))], axis=-1), np.stack([(-1 / B1), (A1 / B1)], axis↩
            ↪ =-1)],
        axis=1)
    Y_path2 = np.stack(
```

```
        [np.stack([(D2 / B2), ((B2 * C2 - A2 * D2) / (B2))], axis=-1), np.stack([(-1 / B2), (A2 / B2)], axis=-1)],
        axis=1)
    Y = Y_path1 + Y_path2
    Y11 = Y[:, 0, 0]
    Y12 = Y[:, 0, 1]
    Y21 = Y[:, 1, 0]
    Y22 = Y[:, 1, 1]
    deltaY = (Y11 + y0) * (Y22 + y0) - Y12 * Y21
    S11 = ((y0 - Y11) * (y0 + Y22) + Y12 * Y21) / (deltaY)
    S12 = (-2 * Y12 * y0) / (deltaY)
    S21 = (-2 * Y21 * y0) / (deltaY)
    S22 = ((y0 + Y11) * (y0 - Y22) + Y12 * Y21) / (deltaY)
    warnings.filterwarnings("default")
    return S11, S12, S21, S22
```

Código E.4: Adaptación de la función unwrap de la librería numpy para valores NaN (not a number)

```
import numpy as np
def unwrap(q, phase):
    # Find NaN's and Inf's
    p = q
    indf = np.isfinite(p)
    # Unwrap finite data (skip non finite entries)
    q[indf] = np.unwrap(p[indf], phase)
    return q
```

Código E.5: Funciones de procesado de señal para caracterizar la respuesta

```
def getLocalZero(delay):
    # Return the local minimum avoiding widths < 1 (maths indeterminacy) and positive delays
    # If out is positive -> Unique or widest and minimum point
    # If out is negative -> Minimum point but not the widest
    # If out is NaN -> No Local Minimum found
    localZero = np.NaN
    peaks, prop = find_peaks(-delay, width=(None, None))
    wdt = prop["widths"]
    erridx = (wdt >= 2) & (delay[peaks] < 0)
    peaks = peaks[erridx]
    if peaks.size == 1:
        localZero = peaks[0]
    elif peaks.size > 1:
        idxW = np.argmax(wdt)
        idxP = np.argmin(delay[peaks])
        if idxW == idxP:
            localZero = peaks[idxP]
        else:
            localZero = -peaks[idxP]
    return localZero


def getNZeros(nZeros, delay):
    iDel = np.repeat(np.NaN, nZeros)
    iDel[0] = getLocalZero(delay[0:999])
    if nZeros >= 2:
        idel = getLocalZero(delay[900:1200])
        iDel[1] = idel + 900 if idel >= 0 else idel - 900
    if nZeros == 3:
        idel = getLocalZero(delay[1001:-1])
        iDel[2] = idel + 1001 if idel >= 0 else idel - 1001
    return iDel

```

```
34
35 def getMeasures(nZeros, delay, SS21, SS11, SS22):
36     warnings.filterwarnings("ignore", message="invalid value encountered in ")
37     warnings.filterwarnings("ignore", message="divide by zero encountered in ")
38     # Get NGD indexes
39     iDel = getNZeros(nZeros, delay)
40     # NGD array for NGD Values
41     Del = np.repeat(np.NaN, nZeros)
42     # Desired S-Parameters
43     S21 = np.repeat(np.NaN, nZeros)
44     S11 = np.repeat(np.NaN, nZeros)
45     S22 = np.repeat(np.NaN, nZeros)
46     # 3 desired bandwidths
47     iBW = np.repeat(np.NaN, 2 * nZeros)
48     iBW1dB = np.repeat(np.NaN, 2 * nZeros)
49     iBW3dB = np.repeat(np.NaN, 2 * nZeros)
50     S21dB = 20 * np.log10(abs(SS21))
51     # Auxiliar for errors
52     err = np.repeat(0, nZeros)
53     for nn, idel in enumerate(iDel):
54         if not np.isnan(idel):
55             if idel >= 0:
56                 Del[nn] = delay[int(idel)]
57                 S21[nn] = np.abs(SS21[int(idel)])
58                 S11[nn] = np.abs(SS11[int(idel)])
59                 S22[nn] = np.abs(SS22[int(idel)])
60             else:
61                 iDel[nn] = np.abs(idel)
62                 Del[nn] = delay[int(iDel[nn])]
63                 S21[nn] = np.abs(SS21[int(iDel[nn])])
64                 S11[nn] = np.abs(SS11[int(iDel[nn])])
65                 S22[nn] = np.abs(SS22[int(iDel[nn])])
66                 err[nn] = -1
67
68             # NGD BANDWIDTH -> iBW
69             # Find positive delays at the left side of NGD point
70             aux, = np.nonzero(delay[0:int(iDel[nn])] >= 0)
71             if aux.size > 0:
72                 # Save last positive delay
73                 iBW[2 * nn] = aux[-1]
74             # Find positive delay at the right side of NGD point
75             aux, = np.nonzero(delay[int(iDel[nn]):] >= 0)
76             if aux.size > 0:
77                 # Save first positive delay
78                 iBW[2 * nn + 1] = aux[0] + int(iDel[nn])
79
80             # 1DB BANDWIDTH -> iBW1dB
81             aux, = np.nonzero(
82                 (S21dB[0:int(iDel[nn])] >= S21dB[int(iDel[nn])] + 1) | (
83                         S21dB[0:int(iDel[nn])] <= S21dB[int(iDel[nn])] - 1))
84             if aux.size > 0:
85                 iBW1dB[2 * nn] = aux[-1]
86             aux, = np.nonzero(
87                 (S21dB[int(iDel[nn]):-1] >= S21dB[int(iDel[nn])] + 1) | (
88                         S21dB[int(iDel[nn]):-1] <= S21dB[int(iDel[nn])] - 1))
89             if aux.size > 0:
90                 iBW1dB[2 * nn + 1] = aux[0] + int(iDel[nn])
91
92             # 3DB BANDWIDTH -> iBW3dB
93             aux, = np.nonzero(
94                 (S21dB[0:int(iDel[nn])] >= S21dB[int(iDel[nn])] + 3) | (
95                         S21dB[0:int(iDel[nn])] <= S21dB[int(iDel[nn])] - 3))
96             if aux.size > 0:
97                 iBW3dB[2 * nn] = aux[-1]
```

```
98              aux, = np.nonzero(
99                  (S21dB[int(iDel[nn]):-1] >= S21dB[int(iDel[nn])] + 3) | (
100                         S21dB[int(iDel[nn]):-1] <= S21dB[int(iDel[nn])] - 3))
101             if aux.size > 0:
102                 iBW3dB[2 * nn + 1] = aux[0] + int(iDel[nn])
103         else:
104             err[nn] = 1
105     warnings.filterwarnings("default")
106     return iDel, Del, S21, S11, S22, iBW, iBW1dB, iBW3dB, err
```